



# SAS/ACCESS® 9.4 for Relational Databases: Reference, Ninth Edition

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS/ACCESS® 9.4 for Relational Databases: Reference, Ninth Edition*. Cary, NC: SAS Institute Inc.

**SAS/ACCESS® 9.4 for Relational Databases: Reference, Ninth Edition**

Copyright © 2016, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

**For a hard copy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

**U.S. Government License Rights; Restricted Rights:** The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

April 2023

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

9.4-P10:acreldb

---

# Contents

<i>What's New in SAS/ACCESS 9.4 for Relational Databases .....</i>	xv
--	----

## PART 1 Concepts 1

<b>Chapter 1 / Overview of SAS/ACCESS Interface to Relational Databases .....</b>	<b>3</b>
About SAS/ACCESS Documentation .....	3
Sample Code .....	4
Methods for Accessing Relational Database Data .....	4
Selecting a SAS/ACCESS Method .....	5
SAS Views of DBMS Data .....	7
Choosing Your Degree of Numeric Precision .....	8
<b>Chapter 2 / Using SAS/ACCESS with SAS Viya .....</b>	<b>13</b>
How SAS/ACCESS Works with the CAS Server .....	13
SAS/ACCESS Usage with SAS Viya .....	15
<b>Chapter 3 / SAS Names and Support for DBMS Names .....</b>	<b>21</b>
DBMS-Specific Naming Conventions .....	22
SAS Naming Conventions .....	22
SAS/ACCESS Default Naming Behaviors .....	24
Renaming DBMS Data .....	25
Options That Affect SAS/ACCESS Naming Behavior .....	25
Naming Behavior When Retrieving DBMS Data .....	26
Naming Behavior When Creating DBMS Objects .....	27
SAS/ACCESS Naming Examples .....	29
<b>Chapter 4 / Data Integrity and Security .....</b>	<b>37</b>
DBMS Security .....	37
SAS Security .....	38
Potential Result Set Differences When Processing Null Data .....	43
<b>Chapter 5 / Performance Considerations .....</b>	<b>47</b>
Increasing Throughput of the SAS Server .....	47
Limiting Retrieval .....	48
Repeatedly Accessing Data .....	49
Sorting DBMS Data .....	49
Temporary Table Support for SAS/ACCESS .....	51
<b>Chapter 6 / Optimizing Your SQL Usage .....</b>	<b>55</b>
Overview: Optimizing Your SQL Usage .....	55
Tracing and Evaluating SQL Generation .....	56
About the DBDIRECTEXEC System Option .....	56
Passing Functions to the DBMS Using PROC SQL .....	58

Date Arithmetic with Queries .....	61
Passing Joins to the DBMS .....	61
When Passing Joins to the DBMS Fails .....	64
Using the DIRECT_SQL= LIBNAME Option .....	65
Passing the WHERE Clause to the DBMS .....	66
Advanced PROC SQL Optimization Hints .....	67
<b>Chapter 7 / Threaded Reads .....</b>	<b>69</b>
Overview: Threaded Reads in SAS/ACCESS .....	69
Underlying Technology of Threaded Reads .....	70
SAS/ACCESS Interfaces and Threaded Reads .....	70
Scope of Threaded Reads .....	71
Options That Affect Threaded Reads .....	71
Generating Trace Information for Threaded Reads .....	72
Performance Impact of Threaded Reads .....	75
Autopartitioning Techniques in SAS/ACCESS .....	76
Data Ordering in SAS/ACCESS .....	77
Two-Pass Processing for SAS Threaded Applications .....	77
When Threaded Reads Do Not Occur .....	78
Summary of Threaded Reads .....	78
<b>Chapter 8 / National Language Support .....</b>	<b>79</b>
Overview: NLS for SAS/ACCESS .....	79
LIBNAME and Other Options for NLS .....	79
Data Types for NLS .....	80
<b>Chapter 9 / How SAS/ACCESS Works .....</b>	<b>81</b>
Introduction to How SAS/ACCESS Works .....	81
How the SAS/ACCESS LIBNAME Statement Works .....	82
How the SQL Pass-Through Facility Works .....	83
How the ACCESS Procedure Works .....	84
How the DBLOAD Procedure Works .....	86
<b>Chapter 10 / In-Database Processing with SAS/ACCESS .....</b>	<b>89</b>
Overview: In-Database Processing .....	89

## PART 2 General Reference 93

<b>Chapter 11 / SAS/ACCESS Features by Host .....</b>	<b>95</b>
Introduction .....	96
Overview of Support for SAS 9.4, SAS Viya 3.5, and Data Connectors .....	96
SAS/ACCESS Interface to Amazon Redshift: Supported Features .....	97
SAS/ACCESS Interface to Aster: Supported Features .....	98
SAS/ACCESS Interface to DB2 under UNIX and PC Hosts: Supported Features .....	99
SAS/ACCESS Interface to DB2 under z/OS: Supported Features .....	100
SAS/ACCESS Interface to Google BigQuery: Supported Features .....	101
SAS/ACCESS Interface to Greenplum: Supported Features .....	102
SAS/ACCESS Interface to Hadoop: Supported Features .....	103
SAS/ACCESS Interface to HAWQ: Supported Features .....	104
SAS/ACCESS Interface to Impala: Supported Features .....	105
SAS/ACCESS Interface to Informix: Supported Features .....	105
SAS/ACCESS Interface to JDBC: Supported Features .....	106

SAS/ACCESS Interface to Microsoft SQL Server: Supported Features .....	107
SAS/ACCESS Interface to MySQL: Supported Features .....	108
SAS/ACCESS Interface to Netezza: Supported Features .....	109
SAS/ACCESS Interface to ODBC: Supported Features .....	110
SAS/ACCESS Interface to OLE DB: Supported Features .....	111
SAS/ACCESS Interface to Oracle: Supported Features .....	112
SAS/ACCESS Interface to PostgreSQL: Supported Features .....	113
SAS/ACCESS Interface to SAP ASE: Supported Features .....	114
SAS/ACCESS Interface to SAP HANA: Supported Features .....	115
SAS/ACCESS Interface to SAP IQ: Supported Features .....	116
SAS/ACCESS Interface to Snowflake: Supported Features .....	117
SAS/ACCESS Interface to Spark: Supported Features .....	118
SAS/ACCESS Interface to Teradata: Supported Features .....	118
SAS/ACCESS Interface to Vertica: Supported Features .....	119
SAS/ACCESS Interface to Yellowbrick: Supported Features .....	120
<b>Chapter 12 / The LIBNAME Statement .....</b>	<b>123</b>
Overview: LIBNAME Statement for Relational Databases .....	127
Assigning a Libref Interactively .....	128
Dictionary .....	129
<b>Chapter 13 / Data Set Options .....</b>	<b>365</b>
About the Data Set Options for Relational Databases .....	370
Dictionary .....	371
<b>Chapter 14 / Macro Variables and System Options .....</b>	<b>655</b>
Introduction to Macro Variables and System Options .....	655
Macro Variables for Relational Databases .....	656
System Options for Relational Databases .....	658
Dictionary .....	659
<b>Chapter 15 / SQL Pass-Through Facility .....</b>	<b>689</b>
About SQL Procedure Interactions .....	689
Syntax: SQL Pass-Through Facility for Relational Databases .....	690
Dictionary .....	691

## PART 3 DBMS-Specific Reference 703

<b>Chapter 16 / SAS/ACCESS Interface to Amazon Redshift .....</b>	<b>705</b>
System Requirements for SAS/ACCESS Interface to Amazon Redshift .....	706
Introduction to SAS/ACCESS Interface to Amazon Redshift .....	706
LIBNAME Statement for the Amazon Redshift Engine .....	707
Data Set Options for Amazon Redshift .....	713
SQL Pass-Through Facility Specifics for Amazon Redshift .....	716
Passing SAS Functions to Amazon Redshift .....	717
Passing Joins to Amazon Redshift .....	718
Bulk Loading for Amazon Redshift .....	718
Encryption with Amazon Redshift .....	721
Locking in the Amazon Redshift Interface .....	721
Working with NLS Characters in Amazon Redshift Data .....	723
Naming Conventions for Amazon Redshift .....	723
Data Types for Amazon Redshift .....	724

Sample Programs for Amazon Redshift .....	726
<b>Chapter 17 / SAS/ACCESS Interface to Aster .....</b>	<b>727</b>
System Requirements for SAS/ACCESS Interface to Aster .....	728
Introduction to SAS/ACCESS Interface to Aster .....	728
LIBNAME Statement for the Aster Engine .....	729
Data Set Options for Aster .....	734
SQL Pass-Through Facility Specifics for Aster .....	736
Autopartitioning Scheme for Aster .....	738
Temporary Table Support for Aster .....	740
Passing SAS Functions to Aster .....	740
Passing Joins to Aster .....	741
Bulk Loading and Unloading for Aster .....	742
Naming Conventions for Aster .....	744
Data Types for Aster .....	745
Sample Programs for Aster .....	747
<b>Chapter 18 / SAS/ACCESS Interface to DB2 under UNIX and PC Hosts .....</b>	<b>749</b>
System Requirements for SAS/ACCESS Interface to DB2 under UNIX and PC Hosts .....	750
Introduction to SAS/ACCESS Interface to DB2 under UNIX and PC Hosts .....	750
LIBNAME Statement for the DB2 Engine under UNIX and PC Hosts .....	751
Data Set Options for DB2 under UNIX and PC Hosts .....	756
SQL Pass-Through Facility Specifics for DB2 under UNIX and PC Hosts .....	759
Autopartitioning Scheme for DB2 under UNIX and PC Hosts .....	761
Temporary Table Support for DB2 under UNIX and PC Hosts .....	764
Calling Stored Procedures in DB2 under UNIX and PC Hosts .....	765
DBLOAD Procedure Specifics for DB2 under UNIX and PC Hosts .....	768
Passing SAS Functions to DB2 under UNIX and PC Hosts .....	770
Passing Joins to DB2 under UNIX and PC Hosts .....	771
Sorting Data That Contains NULL Values .....	772
Bulk Loading for DB2 under UNIX and PC Hosts .....	772
Locking in the DB2 under UNIX and PC Hosts Interface .....	776
Naming Conventions for DB2 under UNIX and PC Hosts .....	777
Data Types for DB2 under UNIX and PC Hosts .....	778
Sample Programs for DB2 under UNIX and PC Hosts .....	783
<b>Chapter 19 / SAS/ACCESS Interface to DB2 under z/OS .....</b>	<b>785</b>
System Requirements for SAS/ACCESS Interface to DB2 under z/OS .....	787
Introduction to SAS/ACCESS Interface to DB2 under z/OS .....	787
LIBNAME Statement for the DB2 Engine under z/OS .....	788
Data Set Options for DB2 under z/OS .....	792
SQL Pass-Through Facility Specifics for DB2 under z/OS .....	795
Autopartitioning Scheme for DB2 under z/OS .....	797
Temporary Table Support for DB2 under z/OS .....	799
Calling Stored Procedures in DB2 under z/OS .....	799
ACCESS Procedure Specifics for DB2 under z/OS .....	802
DBLOAD Procedure Specifics for DB2 under z/OS .....	804
The DB2EXT Procedure .....	806
The DB2UTIL Procedure .....	808
Maximizing DB2 under z/OS Performance .....	814
Passing SAS Functions to DB2 under z/OS .....	817
Passing Joins to DB2 under z/OS .....	819
SAS System Options, Settings, and Macros for DB2 under z/OS .....	819
Bulk Loading for DB2 under z/OS .....	822
Locking in the DB2 under z/OS Interface .....	828

Naming Conventions for DB2 under z/OS .....	829
Data Types for DB2 under z/OS .....	830
Temporal Data for DB2 under z/OS .....	836
Understanding DB2 under z/OS Client/Server Authorization .....	838
DB2 under z/OS Information for the Database Administrator .....	841
Sample Programs for DB2 under z/OS Hosts .....	845
<b>Chapter 20 / SAS/ACCESS Interface to Google BigQuery .....</b>	<b>847</b>
System Requirements for SAS/ACCESS Interface to Google BigQuery .....	848
Introduction to SAS/ACCESS Interface to Google BigQuery .....	848
LIBNAME Statement for the Google BigQuery Engine .....	849
Data Set Options for Google BigQuery .....	853
Connecting to Google BigQuery Using OAuth Authentication .....	854
SQL Pass-Through Facility Specifics for Google BigQuery .....	857
Known Limitations When Working with Google BigQuery .....	858
Working with Views for Google BigQuery .....	859
Updating and Deleting Google BigQuery Data .....	859
Passing SAS Functions to Google BigQuery .....	860
Passing Joins to Google BigQuery .....	862
Bulk Loading and Bulk Unloading for Google BigQuery .....	862
Naming Conventions for Google BigQuery .....	863
Data Types and Conversions for Google BigQuery .....	864
<b>Chapter 21 / SAS/ACCESS Interface to Greenplum .....</b>	<b>869</b>
System Requirements for SAS/ACCESS Interface to Greenplum .....	870
Introduction to SAS/ACCESS Interface to Greenplum .....	870
LIBNAME Statement for the Greenplum Engine .....	871
Data Set Options for Greenplum .....	875
SQL Pass-Through Facility Specifics for Greenplum .....	878
Autopartitioning Scheme for Greenplum .....	880
Temporary Table Support for Greenplum .....	882
Passing SAS Functions to Greenplum .....	883
Passing Joins to Greenplum .....	884
Sorting Data That Contains NULL Values .....	884
Bulk Loading for Greenplum .....	885
Locking in the Greenplum Interface .....	889
Naming Conventions for Greenplum .....	890
Data Types for Greenplum .....	891
Sample Programs for Greenplum .....	894
<b>Chapter 22 / SAS/ACCESS Interface to Hadoop .....</b>	<b>895</b>
System Requirements for SAS/ACCESS Interface to Hadoop .....	896
Introduction to SAS/ACCESS Interface to Hadoop .....	896
LIBNAME Statement for the Hadoop Engine .....	898
Data Set Options for Hadoop .....	906
SQL Pass-Through Facility Specifics for Hadoop .....	908
Temporary Table Support for Hadoop .....	909
Passing SAS Functions to Hadoop .....	909
Passing Joins to Hadoop .....	910
Bulk Loading for Hadoop .....	911
Naming Conventions for SAS and Hive .....	912
Data Types for Hadoop .....	913
Sample Programs for Hadoop .....	922
<b>Chapter 23 / SAS/ACCESS Interface to HAWQ .....</b>	<b>925</b>
System Requirements for SAS/ACCESS Interface to HAWQ .....	926

Introduction to SAS/ACCESS Interface to HAWQ .....	926
LIBNAME Statement for the HAWQ Engine .....	927
Data Set Options for HAWQ .....	931
SQL Pass-Through Facility Specifics for HAWQ .....	934
Autopartitioning Scheme for HAWQ .....	936
Passing SAS Functions to HAWQ .....	938
Passing Joins to HAWQ .....	939
Sorting Data That Contains NULL Values .....	940
Bulk Loading for HAWQ .....	940
Locking in the HAWQ Interface .....	944
Naming Conventions for HAWQ .....	946
Data Types for HAWQ .....	946
Sample Programs for HAWQ .....	949
<b>Chapter 24 / SAS/ACCESS Interface to Impala .....</b>	<b>951</b>
System Requirements for SAS/ACCESS Interface to Impala .....	952
Introduction to SAS/ACCESS Interface to Impala .....	952
LIBNAME Statement for the Impala Engine .....	953
Data Set Options for Impala .....	957
SQL Pass-Through Facility Specifics for Impala .....	959
Passing SAS Functions to Impala .....	960
Passing Joins to Impala .....	961
Sorting Data That Contains NULL Values .....	961
Bulk Loading for Impala .....	962
Naming Conventions for Impala .....	965
Data Types for Impala .....	966
Sample Programs for Impala .....	969
<b>Chapter 25 / SAS/ACCESS Interface to Informix .....</b>	<b>971</b>
System Requirements for SAS/ACCESS Interface to Informix .....	972
Introduction to SAS/ACCESS Interface to Informix .....	972
LIBNAME Statement for the Informix Engine .....	973
Data Set Options for Informix .....	976
SQL Pass-Through Facility Specifics for Informix .....	977
Autopartitioning Scheme for Informix .....	980
Temporary Table Support for Informix .....	982
Passing SAS Functions to Informix .....	982
Passing Joins to Informix .....	983
Locking in the Informix Interface .....	983
Naming Conventions for Informix .....	984
Data Types for Informix .....	985
Informix Servers .....	988
<b>Chapter 26 / SAS/ACCESS Interface to JDBC .....</b>	<b>991</b>
System Requirements for SAS/ACCESS Interface to JDBC .....	992
Introduction to SAS/ACCESS Interface to JDBC .....	992
LIBNAME Statement for the JDBC Engine .....	994
Data Set Options for JDBC .....	998
SQL Pass-Through Facility Specifics for JDBC .....	999
Temporary Table Support for JDBC .....	1001
Passing SAS Functions to JDBC .....	1001
Passing Joins to JDBC .....	1002
Bulk Loading for JDBC .....	1002
Naming Conventions for JDBC .....	1003
Data Types for JDBC .....	1004

<b>Chapter 27 / SAS/ACCESS Interface to Microsoft SQL Server .....</b>	<b>1007</b>
System Requirements for SAS/ACCESS Interface to Microsoft SQL Server .....	1008
Introduction to SAS/ACCESS Interface to Microsoft SQL Server .....	1008
LIBNAME Statement for the Microsoft SQL Server Engine .....	1009
Data Set Options for Microsoft SQL Server .....	1015
SQL Pass-Through Facility Specifics for Microsoft SQL Server .....	1018
Passing Joins to Microsoft SQL Server .....	1019
Passing SAS Functions to Microsoft SQL Server .....	1020
DBLOAD Procedure Specifics for Microsoft SQL Server .....	1021
Temporary Table Support for Microsoft SQL Server .....	1022
Locking in the Microsoft SQL Server Interface .....	1023
Bulk Loading with Microsoft SQL Server .....	1024
Naming Conventions for Microsoft SQL Server .....	1026
Data Types for Microsoft SQL Server .....	1027
Sample Programs for Microsoft SQL Server .....	1030
<b>Chapter 28 / SAS/ACCESS Interface to MySQL .....</b>	<b>1031</b>
System Requirements for SAS/ACCESS Interface to MySQL .....	1032
Introduction to SAS/ACCESS Interface to MySQL .....	1032
LIBNAME Statement for the MySQL Engine .....	1033
Data Set Options for MySQL .....	1036
SQL Pass-Through Facility Specifics for MySQL .....	1038
Autocommit and Table Types .....	1039
Understanding MySQL Update and Delete Rules .....	1040
Temporary Table Support for MySQL .....	1040
Passing SAS Functions to MySQL .....	1041
Passing Joins to MySQL .....	1042
Bulk Loading for MySQL .....	1042
Naming Conventions for MySQL .....	1043
Case Sensitivity for MySQL .....	1044
Data Types for MySQL .....	1044
Sample Programs for MySQL .....	1048
<b>Chapter 29 / SAS/ACCESS Interface to Netezza .....</b>	<b>1049</b>
System Requirements for SAS/ACCESS Interface to Netezza .....	1050
Introduction to SAS/ACCESS Interface to Netezza .....	1050
LIBNAME Statement for the Netezza Engine .....	1050
Data Set Options for Netezza .....	1055
SQL Pass-Through Facility Specifics for Netezza .....	1057
Temporary Table Support for Netezza .....	1059
Passing SAS Functions to Netezza .....	1059
Passing Joins to Netezza .....	1060
Bulk Loading and Unloading for Netezza .....	1061
Naming Conventions for Netezza .....	1063
Data Types for Netezza .....	1064
Sample Programs for Netezza .....	1068
<b>Chapter 30 / SAS/ACCESS Interface to ODBC .....</b>	<b>1069</b>
System Requirements for SAS/ACCESS Interface to ODBC .....	1070
Introduction to SAS/ACCESS Interface to ODBC .....	1070
LIBNAME Statement for the ODBC Engine .....	1075
Data Set Options for ODBC .....	1081
SQL Pass-Through Facility Specifics for ODBC .....	1083
Autopartitioning Scheme for ODBC .....	1087
DBLOAD Procedure Specifics for ODBC .....	1092
Temporary Table Support for ODBC .....	1094

Passing SAS Functions to ODBC .....	1094
Passing Joins to ODBC .....	1095
Bulk Loading for ODBC .....	1096
Locking in the ODBC Interface .....	1096
Naming Conventions for ODBC .....	1098
Data Types for ODBC .....	1098
Sample Programs for ODBC .....	1101
<b>Chapter 31 / SAS/ACCESS Interface to OLE DB .....</b>	<b>1103</b>
System Requirements for SAS/ACCESS Interface to OLE DB .....	1104
Introduction to SAS/ACCESS Interface to OLE DB .....	1104
LIBNAME Statement for the OLE DB Engine .....	1104
Data Set Options for OLE DB .....	1112
SQL Pass-Through Facility Specifics for OLE DB .....	1114
Temporary Table Support for OLE DB .....	1119
Passing SAS Functions to OLE DB .....	1119
Passing Joins to OLE DB .....	1120
Bulk Loading for OLE DB .....	1121
Locking in the OLE DB Interface .....	1121
Accessing OLE DB for OLAP Data .....	1123
Naming Conventions for OLE DB .....	1125
Data Types for OLE DB .....	1126
Sample Programs for OLE DB .....	1129
<b>Chapter 32 / SAS/ACCESS Interface to Oracle .....</b>	<b>1131</b>
System Requirements for SAS/ACCESS Interface to Oracle .....	1132
Introduction to SAS/ACCESS Interface to Oracle .....	1132
LIBNAME Statement for the Oracle Engine .....	1133
Data Set Options for Oracle .....	1139
SQL Pass-Through Facility Specifics for Oracle .....	1143
Autopartitioning Scheme for Oracle .....	1145
ACCESS Procedure Specifics for Oracle .....	1148
DBLOAD Procedure Specifics for Oracle .....	1150
Maximizing Oracle Performance .....	1152
Temporary Table Support for Oracle .....	1152
Passing SAS Functions to Oracle .....	1152
Passing Joins to Oracle .....	1153
Sorting Data That Contains NULL Values .....	1154
Bulk Loading for Oracle .....	1154
Locking in the Oracle Interface .....	1158
Naming Conventions for Oracle .....	1159
Data Types for Oracle .....	1160
Sample Programs for Oracle .....	1169
<b>Chapter 33 / SAS/ACCESS Interface to PostgreSQL .....</b>	<b>1171</b>
System Requirements for SAS/ACCESS Interface to PostgreSQL .....	1172
Introduction to SAS/ACCESS Interface to PostgreSQL .....	1172
LIBNAME Statement for the PostgreSQL Engine .....	1173
Data Set Options for PostgreSQL .....	1178
Best Practices When You Connect to CockroachDB .....	1180
SQL Pass-Through Facility Specifics for PostgreSQL .....	1181
Temporary Table Support for PostgreSQL .....	1182
Running PROC COPY In-Database for PostgreSQL .....	1182
Passing SAS Functions to PostgreSQL .....	1184
Passing Joins to PostgreSQL .....	1185
Bulk Loading and Unloading for PostgreSQL .....	1186

Improve Performance When Deleting or Updating Rows .....	1187
Locking in the PostgreSQL Interface .....	1188
Working with NLS Characters in PostgreSQL Data .....	1189
Naming Conventions for PostgreSQL .....	1189
Data Types for PostgreSQL .....	1190
Sample Programs for PostgreSQL .....	1194
<b>Chapter 34 / SAS/ACCESS Interface to SAP ASE .....</b>	<b>1195</b>
System Requirements for SAS/ACCESS Interface to SAP ASE .....	1196
Introduction to SAS/ACCESS Interface to SAP ASE .....	1196
LIBNAME Statement for the SAP ASE Engine .....	1197
Data Set Options for SAP ASE .....	1201
SQL Pass-Through Facility Specifics for SAP ASE .....	1203
Autopartitioning Scheme for SAP ASE .....	1204
Temporary Table Support for SAP ASE .....	1206
ACCESS Procedure Specifics for SAP ASE .....	1206
DBLOAD Procedure Specifics for SAP ASE .....	1208
Passing SAS Functions to SAP ASE .....	1210
Passing Joins to SAP ASE .....	1210
Bulk Loading for SAP ASE .....	1211
Reading Multiple SAP ASE Tables .....	1212
Locking in the SAP ASE Interface .....	1213
Naming Conventions for SAP ASE .....	1214
Case Sensitivity in SAP ASE .....	1215
Data Types for SAP ASE .....	1216
National Language Support for SAP ASE .....	1222
<b>Chapter 35 / SAS/ACCESS Interface to SAP HANA .....</b>	<b>1223</b>
System Requirements for SAS/ACCESS Interface to SAP HANA .....	1224
Introduction to SAS/ACCESS Interface to SAP HANA .....	1224
LIBNAME Statement for the SAP HANA Engine .....	1225
Data Set Options for SAP HANA .....	1231
SQL Pass-Through Facility Specifics for SAP HANA .....	1234
Understanding SAP HANA Update and Delete Rules .....	1235
Autopartitioning Scheme for SAP HANA .....	1236
SAP HANA Schema Flexibility .....	1237
Temporary Table Support for SAP HANA .....	1238
Passing SAS Functions to SAP HANA .....	1238
Passing Joins to SAP HANA .....	1239
Bulk Loading for SAP HANA .....	1240
Naming Conventions for SAP HANA .....	1241
Data Types for SAP HANA .....	1242
<b>Chapter 36 / SAS/ACCESS Interface to SAP IQ .....</b>	<b>1249</b>
System Requirements for SAS/ACCESS Interface to SAP IQ .....	1250
Introduction to SAS/ACCESS Interface to SAP IQ .....	1250
LIBNAME Statement for the SAP IQ Engine .....	1251
Data Set Options for SAP IQ .....	1256
SQL Pass-Through Facility Specifics for SAP IQ .....	1258
Autopartitioning Scheme for SAP IQ .....	1260
Temporary Table Support for SAP IQ .....	1262
Passing SAS Functions to SAP IQ .....	1262
Passing Joins to SAP IQ .....	1263
Bulk Loading for SAP IQ .....	1263
Locking in the SAP IQ Interface .....	1265
Naming Conventions for SAP IQ .....	1266

Data Types for SAP IQ .....	1267
<b>Chapter 37 / SAS/ACCESS Interface to Snowflake .....</b>	<b>1271</b>
System Requirements for SAS/ACCESS Interface to Snowflake .....	1272
Introduction to SAS/ACCESS Interface to Snowflake .....	1272
LIBNAME Statement for the Snowflake Engine .....	1272
Data Set Options for Snowflake .....	1278
SQL Pass-Through Facility Specifics for Snowflake .....	1280
Understanding Snowflake Update and Delete Rules .....	1282
Temporary Table Support for Snowflake .....	1283
Passing SAS Functions to Snowflake .....	1283
Passing Joins to Snowflake .....	1284
Bulk Loading and Unloading for Snowflake .....	1285
Naming Conventions for Snowflake .....	1288
Data Types for Snowflake .....	1289
<b>Chapter 38 / SAS/ACCESS Interface to Spark .....</b>	<b>1297</b>
System Requirements for SAS/ACCESS Interface to Spark .....	1298
Introduction to SAS/ACCESS Interface to Spark .....	1298
LIBNAME Statement for the Spark Engine .....	1299
Data Set Options for Spark .....	1305
SQL Pass-Through Facility Specifics for Spark .....	1306
Passing SAS Functions to Spark .....	1308
Passing Joins to Spark .....	1309
Bulk Loading for Spark .....	1309
Naming Conventions for SAS and Spark .....	1311
Data Types for Spark .....	1312
Sample Programs for Spark .....	1315
<b>Chapter 39 / SAS/ACCESS Interface to Teradata .....</b>	<b>1319</b>
System Requirements for SAS/ACCESS Interface to Teradata .....	1321
Introduction to SAS/ACCESS Interface to Teradata .....	1321
LIBNAME Statement for the Teradata Engine .....	1323
Data Set Options for Teradata .....	1328
SQL Pass-Through Facility Specifics for Teradata .....	1332
Temporary Table Support for Teradata .....	1335
Passing SAS Functions to Teradata .....	1336
Passing Joins to Teradata .....	1337
Maximizing Teradata Load and Read Performance .....	1337
Autopartitioning Scheme for Teradata (Legacy) .....	1350
Teradata Processing Tips for SAS Users .....	1354
Security: Authentication with Teradata .....	1358
Locking in the Teradata Interface .....	1363
Naming Conventions for Teradata .....	1369
Data Types for Teradata .....	1371
Temporal Data Types for Teradata .....	1377
Sample Programs for Teradata .....	1380
<b>Chapter 40 / SAS/ACCESS Interface to Vertica .....</b>	<b>1383</b>
System Requirements for SAS/ACCESS Interface to Vertica .....	1383
Introduction to SAS/ACCESS Interface to Vertica .....	1384
LIBNAME Statement for the Vertica Engine .....	1384
Data Set Options for Vertica .....	1389
SQL Pass-Through Facility Specifics for Vertica .....	1391
Understanding Vertica Update and Delete Rules .....	1392
Temporary Table Support for Vertica .....	1393

Passing SAS Functions to Vertica .....	1394
Passing Joins to Vertica .....	1395
Locking in the Vertica Interface .....	1395
Naming Conventions for Vertica .....	1396
Data Types for Vertica .....	1397
<b>Chapter 41 / SAS/ACCESS Interface to Yellowbrick .....</b>	<b>1399</b>
System Requirements for SAS/ACCESS Interface to Yellowbrick .....	1400
Introduction to SAS/ACCESS Interface to Yellowbrick .....	1400
LIBNAME Statement for the Yellowbrick Engine .....	1400
Data Set Options for Yellowbrick .....	1405
Known Limitation When Working with Yellowbrick .....	1407
SQL Pass-Through Facility Specifics for Yellowbrick .....	1408
Temporary Table Support for Yellowbrick .....	1408
Passing SAS Functions to Yellowbrick .....	1409
Passing Joins to Yellowbrick .....	1410
Bulk Loading and Unloading for Yellowbrick .....	1410
Locking in the Yellowbrick Interface .....	1412
Naming Conventions for Yellowbrick .....	1413
Data Types for Yellowbrick .....	1414

PART 4 Appendixes 1419



# What's New in SAS/ACCESS 9.4 for Relational Databases

---

## Overview

Here are the new features and enhancements in SAS 9.4 for SAS/ACCESS for Relational Databases.

- new interfaces: PostgreSQL, SAP HANA, and Vertica.
- Platform update: SAS/ACCESS no longer supports HP-UX for PA-RISC architecture.

In [SAS 9.4M1](#), the BL\_ESCAPE= data set option is new for multiple interfaces.

In [SAS 9.4M2](#), Cloudera Impala is a new SAS/ACCESS interface.

In [SAS 9.4M3](#), these features are new or enhanced:

- HAWQ is a new SAS/ACCESS interface.
- The format for specifying octal codes using the BL\_ESCAPE= data set option has changed.
- Documentation: A new section about [National Language Support \(NLS\)](#) considerations, issues, and limitations was added.

In [SAS 9.4M5](#), these features are new or enhanced:

- SAS/ACCESS supports 64-bit HP-UX on an Itanium processor. SAS/ACCESS also no longer supports Linux for 32-bit Intel architecture.
- Temporary files that are created during bulk loading are stored by default in the temporary file location that is specified by the UTILLOC= system option, instead of in the current directory.

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS and SAS Viya.

In [SAS Viya 3.4](#), support for the JDBC interface on SAS Viya 3.4 was added.

In [SAS 9.4M6](#), support for the JDBC interface on SAS 9.4 was added.

In the August 2019 release of SAS/ACCESS on [SAS 9.4M6](#), support for the Google BigQuery and Snowflake interfaces was added.

In [SAS Viya 3.5](#), sample programs were moved from an installation directory onto a GitHub repository. Samples for most interfaces are available from <https://github.com/sassoftware/sas-access-samples>

In [SAS 9.4M7](#), support was added for the Spark interface and for the Yellowbrick interface. Also in this release, SAS/ACCESS Interface to Spark is no longer available on SAS Viya 3.4 or SAS Viya 3.5.

---

# System Options

---

## DBIDIRECTEXEC

In [SAS 94M8](#), the DBIDIRECTEXEC system option is enabled by default for all data sources. This is a change for these data sources:

DB2 under UNIX and PC Hosts	Netezza
DB2 under z/OS	Oracle
Greenplum	PostgreSQL
Hadoop	SAP ASE
Impala	SAP HANA
Informix	SAP IQ
Microsoft SQL Server	Teradata
MySQL	Vertica

---

## SQLGENERATION=

In [SAS 9.4M3](#), the default value of the SQLGENERATION= system option became '(NONE DBMS='TERADATA DB2 ORACLE NETEZZA ASTER GREENPLM HADOOP SAPHANA IMPALA HAWQ').

In [SAS 9.4M4](#), the default value of the SQLGENERATION= system option became '(NONE DBMS='TERADATA DB2 ORACLE NETEZZA ASTER GREENPLM HADOOP SAPHANA IMPALA HAWQ POSTGRES REDSHIFT SQLSVR VERTICA'). The corresponding LIBNAME option also supports these engines.

In [SAS 9.4M7](#), the default value for the SQLGENERATION= system option became '(NONE DBMS='TERADATA DB2 ORACLE NETEZZA ASTER GREENPLM HADOOP SAPHANA IMPALA HAWQ POSTGRES REDSHIFT SQLSVR VERTICA BIGQUERY SNOW YBRICK'). The corresponding LIBNAME option also supports these engines.

In the August 2021 update for [SAS Viya 3.5](#), the default for SQLGENERATION was changed to '(NONE DBMS='TERADATA DB2 ORACLE NETEZZA ASTER GREENPLM HADOOP SAPHANA IMPALA HAWQ POSTGRES REDSHIFT SQLSVR VERTICA SNOW').

In [SAS 9.4M8](#), the default value for the SQLGENERATION= system option was updated to include MYSQL.

---

# SAS/ACCESS Interface to Amazon Redshift

In the April 2016 release of SAS/ACCESS on [SAS 9.4M3](#), support was added for the Amazon Redshift interface.

In [SAS 9.4M4](#), support was added for these features or enhancements:

- The engine name *redshift* is available (instead of *sasiorst*) in the LIBNAME statement.
- Bulk loading using the Amazon S3 tool is available. This includes support for several new data set options:

BL_BUCKET=	BL_REGION=
BL_COMPRESS=	BL_SECRET=
BL_CONFIG=	BL_TOKEN=
BL_DELIMITER=	BL_USE_ESCAPE=
BL_KEY=	BL_USE_MANIFEST=
BL_NUM_DATAFILES=	BL_USE_SSL=

- In-database processing is available for Base SAS procedures FREQ, MEANS, RANK, REPORT, SORT, SUMMARY, and TABULATE.

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS Interface to Amazon Redshift and SAS Viya. This support includes an associated data connector that enables you to load data to the CAS server.

In [SAS Viya 3.4](#), the following changes or enhancements were made:

- Support was added for the following options:
  - BL\_AWS\_CONFIG\_FILE= data set option
  - BL\_AWS\_PROFILE\_NAME= data set option
  - COMPLETE= connection option
  - CURSOR\_TYPE= LIBNAME option and data set option
  - DBNULLKEYS= LIBNAME option and data set option
  - INSERT\_SQL= LIBNAME option
  - KEYSET\_SIZE= LIBNAME option and data set option
  - PROMPT= and NOPROMPT= connection options
  - QUALIFIER= LIBNAME option
- Support for the following options has changed:
  - AUTOCOMMIT= LIBNAME option: The default value is YES.
  - DBCLIENT\_MAX\_BYTES= LIBNAME option: The default value is the length of the longest-byte character for the SAS session encoding.

- DBSERVER\_MAX\_BYTES= LIBNAME option: There is no default value.
- INSERTBUFF= LIBNAME option: The default value is 1.
- PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= LIBNAME options: The default value is NO.
- SCHEMA= LIBNAME option: There is no default value.
- UPDATE\_MULT\_ROWS= LIBNAME option: This option is not supported.

In SAS 9.4M6, support was added for these features:

- server-side encryption with bulk loading or bulk unloading. This feature includes support for the BL\_ENCKEY= LIBNAME and data set options.
- bulk unloading (data retrieval). This feature includes support for the following new options:

BL\_AWS\_CONFIG\_FILE= LIBNAME option  
BL\_AWS\_PROFILE\_NAME= LIBNAME option  
BL\_BUCKET= LIBNAME option  
BL\_COMPRESS= LIBNAME option  
BL\_CONFIG= LIBNAME option  
BL\_DEFAULT\_DIR= LIBNAME option and data set option  
BL\_DELETE\_DATAFILE= LIBNAME= option  
BL\_DELIMITER= LIBNAME option  
BL\_IAM\_ROLE= LIBNAME option and data set option  
BL\_KEY= LIBNAME option  
BL\_NUM\_READ\_THREADS= LIBNAME option and data set option  
BL\_OPTIONS= LIBNAME option  
BL\_REGION= LIBNAME option  
BL\_SECRET= LIBNAME option  
BL\_TOKEN= LIBNAME option  
BL\_USE\_ESCAPE= LIBNAME option  
BL\_USE\_SSL= LIBNAME option  
BULKUNLOAD= LIBNAME option and data set option  
SUB\_CHAR= LIBNAME option

- support for the CONOPTS= LIBNAME option

In SAS 9.4M7, the default value for the INSERTBUFF= LIBNAME option is now 250.

In the April 2022 update to SAS/ACCESS for SAS 9.4M7 and SAS Viya 3.5, the USE\_ODBC\_CL= LIBNAME option is not supported.

In SAS 9.4M8, support was added for the SAS\_REDSHIFT\_UPDATE\_WARNING= environment variable to facilitate working with NLS data that would otherwise trigger errors in SAS.

---

## SAS/ACCESS Interface to Aster

In [SAS 9.4](#), support was added for the `BL_DATAFILE_PATH=` data set option and the `BL_MAPFILE=` data set option.

In [SAS 9.4M1](#), support was added for AIX and Solaris hosts for SPARC.

In [SAS 9.4M2](#), support was added for bulk loading and bulk unloading. This support includes the `BULKUNLOAD= LIBNAME` option and `BULKUNLOAD=` data set option.

In [SAS 9.4M3](#), support for these options was added:

- `POST_STMT_OPTS=` data set and `LIBNAME` options
- `POST_TABLE_OPTS=` data set option
- `PRE_STMT_OPTS=` data set option
- `PRE_TABLE_OPTS=` data set option

In [SAS Viya 3.4](#), support for the `BL_DEFAULT_DIR= LIBNAME` option was added.

In [SAS 9.4M6](#), support for the `CONOPTS= LIBNAME` option was added.

Starting with [SAS 9.4M8](#), SAS/ACCESS Interface to Aster is no longer available. If you have an instance of SAS/ACCESS Interface to Aster and plan to upgrade to SAS 9.4M8 or later, SAS recommends that you unconfigure and uninstall it. For more information, see [Unconfiguring and Uninstalling Retired Products](#).

---

## SAS/ACCESS Interface to DB2 under UNIX and PC Hosts

In [SAS 9.4](#), support was added for the `PROGRAM_NAME= LIBNAME` option for DB2 monitoring.

In [SAS 9.4M1](#), support for these options was added:

- `CHAR_AS_BINARY=` data set option
- `PRESERVE_USER= LIBNAME` option

In [SAS 9.4M2](#), the maximum length for column names is now 32 characters (rather than 30).

In [SAS 9.4M3](#), support for these options was added:

- `POST_STMT_OPTS=` data set and `LIBNAME` options
- `POST_TABLE_OPTS=` data set option
- `PRE_STMT_OPTS=` data set option

- [PRE\\_TABLE\\_OPTS=](#) data set option

In [SAS 9.4M4](#), support for the `SAS_DB2_TS_REDUCE_SCALE` environment variable was added.

In [SAS 9.4M5](#), support was added for these options:

- [DBCLIENT\\_MAX\\_BYTES=](#) data set option
- [DBNULLWHERE= LIBNAME](#) option and the [DBNULLWHERE=](#) data set option
- [DBSERVER\\_MAX\\_BYTES=](#) data set option

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS Interface to DB2 under UNIX and PC Hosts and SAS Viya. This includes support for the DB2 data connector, which enables you to load data to the CAS server.

In [SAS Viya 3.4](#), changes were made to the following options:

- [AUTOCOMMIT LIBNAME](#) option: The default value is NO.
- [CURSOR\\_TYPE= LIBNAME](#) option: There is no default value.
- [DBINDEX= LIBNAME](#) option: The default value is NO.
- [DBSERVER\\_MAX\\_BYTES= LIBNAME](#) option: There is no default value.
- [DBSLICEPARM= LIBNAME](#) option: The default value is NONE.
- [PRESERVE\\_COL\\_NAMES=](#) and [PRESERVE\\_TAB\\_NAMES= LIBNAME](#) options: The default value is YES.
- [READ\\_ISOLATION\\_LEVEL= LIBNAME](#) option: There is no default value.

In [SAS 9.4M6](#), the default value for the [PRESERVE\\_COL\\_NAMES=](#) and [PRESERVE\\_TAB\\_NAMES= LIBNAME](#) options is NO.

In [SAS 9.4M7](#), the [BL\\_METHOD=](#) data set option now supports the values IMPORT and LOAD in addition to CLILOAD.

In the March 2021 update of SAS/ACCESS for [SAS 9.4M7](#), syntax for the [BL\\_OPTIONS=](#) data set option now shows that commas are not required between multiple bulk-load option values.

In [SAS 9.4M8](#), support was added for the [PRESERVE\\_COMMENTS= LIBNAME](#) option and `DB2_SQL_COMMENT` macro variable. These options enable you to pass comments down to your data source in a query.

---

## SAS/ACCESS Interface to DB2 under z/OS

In [SAS 9.4M1](#), support for the [ALLOWED\\_SQLCODES= LIBNAME](#) option was added.

In [SAS 9.4M2](#), these features are new or updated:

- support for the BLOB, CLOB, and DBCLOB data types

- The DB2DBUG system option is no longer supported. To view the same database actions in the log, specify the SASTRACE= system option instead as SASTRACE=' , , , d'.

In [SAS 9.4M3](#), support for these options was added:

- [POST\\_STMT\\_OPTS](#)= [data set](#) and [LIBNAME](#) options
- [POST\\_TABLE\\_OPTS](#)= [data set](#) option
- [PRE\\_STMT\\_OPTS](#)= [data set](#) option
- [PRE\\_TABLE\\_OPTS](#)= [data set](#) option

In [SAS 9.4M6](#), support was added for the ALLOW\_SQLCODES= LIBNAME option.

---

## SAS/ACCESS Interface to Google BigQuery

In the August 2019 release of SAS/ACCESS for [SAS 9.4M6](#), support was added for the Google BigQuery interface.

In the November 2019 release of [SAS 9.4M6](#) and [SAS Viya 3.5](#), support was added for the ARRAY, GEOGRAPHY, NUMERIC, and RECORD data types.

In the April 2020 release of SAS/ACCESS for [SAS 9.4M6](#) and [SAS Viya 3.5](#), the following LIBNAME options were added to support OAuth authentication: CLIENT\_ID=, CLIENT\_SECRET=, and REFRESH\_TOKEN=. The PROXY= LIBNAME option was added to support using a connection proxy. Support was also added for the DBCLIENT\_MAX\_BYTES= LIBNAME option.

Beginning in the May 2020 release of SAS/ACCESS for [SAS 9.4M6](#) and [SAS Viya 3.5](#), the SCHEMA= LIBNAME option is required to access Google BigQuery data. The PROJECT= LIBNAME option is also required.

In [SAS 9.4M7](#), the following enhancements were made:

- support for bulk unloading (data retrieval) into SAS. This add support for the BULKUNLOAD=, BL\_BUCKET=, and BL\_NUM\_READ\_THREADS= LIBNAME and data set options. The BL\_DEFAULT\_DIR= and BL\_DELETE\_DATAFILE= options can also be used for bulk unloading.
- ability to run these procedures in-database: FREQ, MEANS, RANK, REPORT, SORT, SUMMARY, and TABULATE. This support includes support for the SQLGENERATION= LIBNAME option and the SQLGENERATION= system option.

In the March 2021 update for SAS/ACCESS on [SAS 9.4M7](#), a restriction was added for the CRED\_PATH= option. The value of this option must be enclosed in single quotation marks ( ' ).

In the April 2021 update for SAS/ACCESS on [SAS 9.4M7](#) and [SAS Viya 3.5](#), the following updates were made:

- Support was added for large result sets. This includes support for the ALLOW\_LARGE\_RESULTS=, LARGE\_RESULTS\_DATASET=, and LARGE\_RESULTS\_EXPIRATION\_TIME= LIBNAME options.

- Support was added for the READ\_MODE= LIBNAME option and data set option. This option enables you to use the Extractor API or the Storage API to move data into SAS. It is recommended that you set READ\_MODE=STORAGE if the Storage API is available for Google BigQuery.
- Support was added for the SCANSTRINGCOLUMNS= LIBNAME option and data set option. This option enables more efficient handling of VARCHAR values in Google BigQuery.
- Updates were made for data type conversions of incoming BOOLEAN, FLOAT64, and NUMERIC data. An update was made for exporting datetime data to Google BigQuery.

---

## SAS/ACCESS Interface to Greenplum

In SAS 9.4, support was added for the BL\_DATAFILE\_EXISTS= data set option.

In SAS 9.4M2, support was added for these options:

- DBSLICE= data set option
- DBSLICEPARM= LIBNAME option, data set option, and system option

In SAS 9.4M3, these features are new or enhanced:

- You can pass the \*\* operator, COT function, and COMPRESS function to Greenplum via the SQL pass-through facility.
- The POST\_STMT\_OPTS= [data set](#) and [LIBNAME](#) options are available.
- The [POST\\_TABLE\\_OPTS=](#) data set option is available.
- The [PRE\\_STMT\\_OPTS=](#) data set option is available.
- The [PRE\\_TABLE\\_OPTS=](#) data set option is available.

In SAS 9.4M4, support was added for the READ\_LOCK\_TYPE= and UPDATE\_LOCK\_TYPE= LIBNAME options.

In SAS Viya 3.3, support was added for the Greenplum interface in SAS Viya.

In SAS Viya 3.4, the following changes or enhancements were made:

- BULKLOAD= LIBNAME option: Support was added.
- DBINDEX= LIBNAME option: The default value is NO.
- DBNULLKEYS= LIBNAME option: The default value is YES.
- DBPROMPT= LIBNAME option: The default value is NO.
- DBSLICEPARM= LIBNAME option: The default value is NONE.
- KEYSET\_SIZE= data set option: Support was added.
- PRESERVE\_COL\_NAMES= LIBNAME option: The default value is YES.
- READ\_ISOLATION\_LEVEL= data set option: Support was added.
- UPDATE\_ISOLATION\_LEVEL= data set option: Support was added.

In SAS 9.4M6, support was added for the CONOPTS= LIBNAME option.

---

# SAS/ACCESS Interface to Hadoop

In SAS 9.4, support was added for these changes and enhancements:

- COLUMN\_DELIMITER= data set option
- HDFS\_TEMPDIR=, HDFS\_METADIR=, HDFS\_DATADIR=, SUBPROTOCOL=, HIVE\_PRINCIPAL=, and HDFS\_PRINCIPAL= LIBNAME connection options
- SQLGENERATION= LIBNAME and system options: The default values have changed.

In SAS 9.4M2, support was added for these enhancements or changes:

- data types TIMESTAMP, DATE, and VARCHAR for Hive 0.12, and CHAR for Hive 0.13.
- PROPERTIES= LIBNAME option
- the SAS environment variable SAS\_HADOOP\_RESTFUL
- HiveServer2. Hive continues to be supported in this release.
- create and append to non-textual Hive table formats, such as SEQUENCEFILE, RCFILE, ORC, and PARQUET
- Hive authorization and authentication using IBM InfoSphere BigInsights 2.1
- SUBPROTOCOL= connection option: The default value is now HiveServer2.
- The CFG= LIBNAME option is deprecated. Use the SAS environment variable SAS\_HADOOP\_CONFIG\_PATH instead.

In SAS 9.4M3, support was added for these features and changes:

- BINARY and DECIMAL data type support
- CONFIG= and CONFIGDIR= LIBNAME and data set options
- DBCREATE\_TABLE\_OPTS= LIBNAME option
- POST\_STMT\_OPTS=, POST\_TABLE\_OPTS=, PRE\_STMT\_OPTS=, and PRE\_TABLE\_OPTS= data set options
- POST\_STMT\_OPTS= LIBNAME option
- TRANSCODE\_FAIL= LIBNAME and data set options
- SQLGENERATION= LIBNAME and system options
- SQOOP procedure. For more information about the SQOOP procedure, see *Base SAS Procedures Guide*.
- The NUMTASKS= LIBNAME option is no longer supported.

In SAS 9.4M4, these features are new or enhanced:

- ANALYZE= and SCRATCH\_DB= LIBNAME and data set options
- SUBPROTOCOL= LIBNAME connection option: Hive2 is the default and is the only valid value.

- Temporary tables are supported.
- Additional Knox, Kerberos, Sentry, and RecordService security support was added.

In [SAS Viya 3.2](#), integration was added between SAS/ACCESS Interface to Hadoop and SAS Viya. This includes support for a data connector. You can also transfer data in parallel if you have licensed SAS In-Database Technologies.

In [SAS 9.4M5](#), support was added for these features and changes:

- CONNECTION= LIBNAME option: The default value is SHAREDREAD.
- URI= LIBNAME connection option
- KNOX\_GATEWAY\_URL environment variable
- Apache Sentry RecordService
- automatic push-down of the MOD (new) and SCAN functions

In [SAS Viya 3.4](#), support was added for these changes and enhancements:

- BULKLOAD= LIBNAME option and data set options
- DBCONTERM= LIBNAME option
- DBLIBINIT= LIBNAME option
- DBLIBTERM= LIBNAME option
- DBMAX\_TEXT= LIBNAME option: The default value is 1024.
- DBSASLABEL= LIBNAME and data set options
- QUERY\_TIMEOUT= LIBNAME and data set options
- SQL\_FUNCTIONS\_COPY= LIBNAME option

[SAS 9.4M6](#) includes these changes:

- BULKLOAD= data set option: The default value is YES.
- SQL\_FUNCTIONS=: To work with HOUR, MINUTE, and SECOND functions, you must specify SQL\_FUNCTIONS=ALL, as these are no longer passed down automatically.

In [SAS Viya 3.5](#), the TRANSCODE\_FAIL= LIBNAME option is deprecated. Use the SUB\_CHAR= LIBNAME option instead.

In [SAS 9.4M7](#), support was added for the DRIVERCLASS= LIBNAME option.

In the June 2022 update for SAS/ACCESS on [SAS 9.4M7](#), support was added for the SUB\_CHAR= LIBNAME option and the SUB\_CHAR= data set option.

In [SAS 9.4M8](#), the following enhancements and changes were made:

- ability to override the default read buffer size. Support has been added for the READBUFF= LIBNAME and data set options.
- ability to specify the default location to create new external tables. Support has been added for the DBCREATE\_TABLE\_LOCATION= LIBNAME option. This option is used with the DBCREATE\_TABLE\_EXTERNAL= LIBNAME option.
- ability to specify the default file format of the Hadoop bulk load staging file. Support has been added for the BL\_FORMAT= LIBNAME and data set options.
- support for JDBC drivers other than the Apache Hive open source driver. For more information, see [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#).

- ability to specify additional resources to add to the JDBC class path. Support has been added for the CLASSPATH= LIBNAME option.
- The CONFIG= LIBNAME option and the CONFIG= data set option are deprecated. It is preferable to use the documented configuration steps and set the appropriate environment variables. For more information, see [Running the Hadoop Tracer Script](#).
- The DRIVER= LIBNAME option is now an alias for the DRIVERCLASS= LIBNAME option.
- The HDFS\_PRINCIPAL= LIBNAME option is deprecated.
- The HIVE\_PRINCIPAL= LIBNAME option is deprecated.

---

## SAS/ACCESS Interface to HAWQ

In [SAS 9.4M3](#), support was added for the HAWQ interface.

In [SAS 9.4M4](#), support was added for the READ\_LOCK\_TYPE= and UPDATE\_LOCK\_TYPE= LIBNAME options.

In [SAS Viya 3.3](#), support was added for the HAWQ interface in SAS Viya.

In [SAS Viya 3.4](#), support was added for these options:

- BL\_DEFAULT\_DIR= LIBNAME option
- BULKLOAD= LIBNAME option
- KEYSET\_SIZE= data set option
- LOGIN\_TIMEOUT= LIBNAME option
- READ\_ISOLATION\_LEVEL= data set option
- UPDATE\_ISOLATION\_LEVEL= data set option

In [SAS Viya 3.5](#), support for the following options has changed:

- DBNULLKEYS= LIBNAME option: The default value is YES.
- PRESERVE\_COL\_NAME= LIBNAME option: The default value is YES.
- SPOOL= LIBNAME option: The default value is YES.

Starting with [SAS 9.4M8](#), SAS/ACCESS Interface to HAWQ is no longer available. If you have an existing installation of SAS/ACCESS Interface to HAWQ and plan to upgrade to SAS 9.4M8 or later, SAS recommends that you first unconfigure and uninstall SAS/ACCESS Interface to HAWQ. For more information, see [Unconfiguring and Uninstalling Retired Products](#).

As an alternative to HAWQ, consider storing your data on Greenplum and accessing it with SAS/ACCESS Interface to Greenplum.

---

# SAS/ACCESS Interface to Impala

In [SAS 9.4M2](#), support was added for the Impala interface.

In [SAS 9.4M2](#), support was added for these features:

- `SAS_HADOOP_JAR_PATH` environment variable
- `IMPALA_PRINCIPAL=`, `HDFS_PRINCIPAL= LIBNAME` option, and the `HDFS_PRINCIPAL=` data set option
- `PARTITIONED_BY=` data set option

In [SAS 9.4M2](#), you can use an environment variable and these options for bulk loading:

- `SAS_HADOOP_RESTFUL` environment variable: Set `SAS_HADOOP_RESTFUL` to 1.
- `BL_HOST=`, `BL_PORT=`, and `BULKLOAD= LIBNAME` options
- `BL_DATAFILE=`, `BL_DELETE_DATAFILE=`, `BL_HOST=`, `BL_PORT=`, and `BULKLOAD=` data set options

In [SAS 9.4M3](#), these features are new or enhanced:

- Use `impala` in the `LIBNAME` statement to specify this engine.
- `CONFIG=` and `CONFIGDIR= LIBNAME` and data set options
- `DBCLIENT_MAX_BYTES=` and `DBSERVER_MAX_BYTES= LIBNAME` options
- `DRIVER_VENDOR= LIBNAME` option
- `POST_STMT_OPTS=`, `POST_TABLE_OPTS=`, `PRE_STMT_OPTS=`, and `PRE_TABLE_OPTS=` data set options.
- `POST_STMT_OPTS= LIBNAME` option
- `SQLGENERATION= LIBNAME` and system options

[SAS 9.4M5](#) supports these features:

- Apache Sentry RecordService
- DECIMAL data type

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS Interface to Impala and SAS Viya. This integration includes a data connector to load data to the CAS server.

In [SAS Viya 3.4](#), support was added for these options:

- `CURSOR_TYPE= LIBNAME` and data set options
- `DBCLIENT_MAX_BYTES= LIBNAME` option
- `DBCOMMIT= LIBNAME` and data set options
- `DBINDEX= LIBNAME` option
- `DBSERVER_MAX_BYTES= LIBNME` option

- `DELETE_MULT_ROWS= LIBNAME` option
- `PRESERVE_COL_NAME= LIBNAME` and data set options
- `PRESERVE_TAB_NAMES= LIBNAME` option
- `SQL_FUNCTIONS_COPY= LIBNAME` option
- `UPDATE_MULT_ROWS= LIBNAME` option

In [SAS 9.4M6](#), support was added for the `CONOPTS= LIBNAME=` option.

In [SAS Viya 3.5](#), support for the `SCRATCH_DB= LIBNAME` and data set option was added.

In the May 2020 update for SAS/ACCESS on [SAS 9.4M6](#) and [SAS Viya 3.5](#), the default value for the `DBCLIENT_MAX_BYTES= LIBNAME` option now matches the maximum number of bytes per single character of the SAS session encoding.

In SAS 9.4M8, the `CONFIG= LIBNAME` option and the `CONFIG=` data set option are deprecated. It is preferable to use the tracer tool to configure the appropriate environment variables. For more information, see [Running the Hadoop Tracer Script](#).

---

## SAS/ACCESS Interface to Informix

In [SAS 9.4M3](#), these options were added:

- `POST_STMT_OPTS=` [data set](#) and `LIBNAME` options
- `POST_TABLE_OPTS=` data set option
- `PRE_STMT_OPTS=` data set option
- `PRE_TABLE_OPTS=` data set option

---

## SAS/ACCESS Interface to JDBC

In [SAS Viya 3.4](#), support was added for the JDBC interface.

In [SAS 9.4M6](#), support was added for the JDBC interface.

In [SAS Viya 3.5](#), the `TRANSCODE_FAIL= LIBNAME` option is deprecated. Use the `SUB_CHAR= LIBNAME` option instead.

In [SAS 9.4M8](#), the following changes and enhancements were made:

- support for enhanced transaction logging. This is controlled with the `DRIVER_TRACE=` and related options.
- ability to override the default read buffer size. Support has been added for the `READBUFF= LIBNAME` and data set options.

# SAS/ACCESS Interface to Microsoft SQL Server

In [SAS 9.4M3](#), support was added for these features:

- 32-bit Microsoft Windows and 64-bit Microsoft Windows platforms
- POST\_STMT\_OPTS=, POST\_TABLE\_OPTS=, PRE\_STMT\_OPTS=, and PRE\_TABLE\_OPTS= data set options
- POST\_STMT\_OPTS= LIBNAME option

In [SAS 9.4M4](#), these features are new or enhanced:

- Ability to connect to the Microsoft Azure SQL Database
- Support for in-database processing of these Base SAS procedures: FREQ, MEANS, RANK, REPORT, SORT, SUMMARY, and TABULATE

[SAS 9.4M5](#) adds support for the DBNULLWHERE= LIBNAME option and the DBNULLWHERE= data set option.

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS Interface to Microsoft SQL Server and SAS Viya. This integration includes a data connector to load data to the CAS server.

In [SAS Viya 3.4](#), the following changes and enhancements were made:

- Support was added for the following options:
  - DATETIME2= LIBNAME option and data set option
  - LOGIN\_TIMEOUT= LIBNAME option
  - SQL\_FUNCTIONS= LIBNAME option
  - SQL\_FUNCTIONS\_COPY= LIBNAME option
  - WARN\_BIGINT= LIBNAME option
- Default values were modified for these options:
  - AUTOCOMMIT= LIBNAME option: The default value is NO.
  - DBINDEX= LIBNAME option and data set option: The default value is NO.
  - DBSLICEPARM= LIBNAME option and data set option: The default value is NONE.

In [SAS 9.4M7](#), support was added for connecting to and bulk loading to Azure Synapse Analytics when using a Microsoft Azure Data Lake Storage Gen 2 storage account. Connecting to Azure Synapse Analytics requires that you specify values for the AZUREAUTHCACHELOC= and AZURETENANTID= system options. Support for bulk loading includes these options:

- BL\_ACCOUNTNAME= LIBNAME option and data set option
- BL\_APPLICATIONID= LIBNAME option and data set option
- BL\_COMPRESS= LIBNAME option and data set option

BL\_DEFAULT\_DIR= LIBNAME option and data set option  
BL\_DELETE\_DATAFILE= LIBNAME option and data set option  
BL\_DELIMITER= LIBNAME option and data set option  
BL\_DNSSUFFIX= LIBNAME option and data set option  
BL\_FILESYSTEM= LIBNAME option and data set option  
BL\_FOLDER= LIBNAME option and data set option  
BL\_IDENTITY= LIBNAME option and data set option  
BL\_LOG= LIBNAME option and data set option  
BL\_MAXERRORS= LIBNAME option and data set option  
BL\_NUM\_DATAFILES= data set option  
BL\_OPTIONS= LIBNAME option and data set option  
BL\_SECRET= LIBNAME option and data set option  
BL\_TIMEOUT= LIBNAME option and data set option  
BL\_USE\_ESCAPE= LIBNAME option and data set option  
BL\_USE\_LOG= LIBNAME option and data set option  
BULKLOAD= LIBNAME option and data set option

In SAS 9.4M8, support was added for the PRESERVE\_COMMENTS= LIBNAME option and SQLSERVER\_SQL\_COMMENT macro variable. These options enable you to pass comments down to your data source in a query.

---

## SAS/ACCESS Interface to MySQL

In SAS 9.4, support was added for bulk loading.

In SAS 9.4M3, these features are new or enhanced:

- ATAN2 and TRANWRD functions: You can now pass these functions to MySQL.
- The list of functions that can be passed automatically to MySQL and the list of functions that require SQL\_FUNCTIONS=ALL have been updated.

In SAS 9.4M4, support was added for these LIBNAME options:

- SSL\_CA=
- SSL\_CERT=
- SSL\_CIPHER=
- SSL\_KEY=

In SAS Viya 3.3, support was added for the MySQL interface in SAS Viya.

In SAS Viya 3.4, the following changes and enhancements were made:

- Support was added for the SAS Data Connector to MySQL. The data connector enables you to transfer large amounts of data between your MySQL database and the CAS server.
- Support for these LIBNAME options was added:
  - BULKLOAD=

- DBCLIENT\_MAX\_BYTES=
- SCHEMA=

In [SAS Viya 3.5](#), support was added for the JSON data type.

In [SAS 9.4M7](#), the default value for the INSERTBUFF= LIBNAME option has changed from 0 to 1.

In the September 2022 update for SAS/ACCESS for [SAS 9.4M7](#) and [SAS Viya 3.5](#), a restriction was modified that concerns using the MySQL interface to access SingleStore (formerly MemSQL) data sources. To access SingleStore, specify UTF-8 as your session encoding.

In [SAS 9.4M8](#), support was added for the following features and enhancements:

- in-database processing of the FREQ, MEANS, RANK, REPORT, SORT, SUMMARY, and TABULATE procedures.
- The default value for the SQLGENERATION= system option was updated to include MYSQL so that in-database processing is enabled automatically.
- There is no longer support for SAS/ACCESS Interface to MySQL on the AIX platform. Documentation remains for existing customers.

---

## SAS/ACCESS Interface to Netezza

In [SAS 9.4](#), support was added for the SYNONYMS= LIBNAME option.

In [SAS 9.4M2](#), support was added for the following features:

- ST\_GeOMETRY and VARBINARY data types
- multiple Netezza schemas

In [SAS 9.4M3](#), these options were added:

- POST\_STMT\_OPTS= [data set](#) and [LIBNAME](#) options
- [POST\\_TABLE\\_OPTS](#)= data set option
- [PRE\\_STMT\\_OPTS](#)= data set option
- [PRE\\_TABLE\\_OPTS](#)= data set option

In [SAS 9.4M4](#), support was added for the PRESERVE\_USER= LIBNAME option. A corresponding environment variable, [SAS\\_NETEZZA\\_PRESERVE\\_USER](#), was also added.

In the April 2016 release for SAS/ACCESS on SAS 9.4, support for SAS/ACCESS Interface to Netezza on the Solaris for x64 platform is new.

[SAS 9.4M5](#) adds support for the BL\_DEFAULT\_DIR= data set option.

In [SAS Viya 3.3](#), support was added for the Netezza interface in SAS Viya.

In [SAS Viya 3.4](#), these changes or enhancements were made:

- DBINDEX= LIBNAME option and data set option: The default value is NO.
- DELETE\_MULT\_ROWS= LIBNAME option: This option is no longer supported.

- PRESERVE\_TAB\_NAMES= LIBNAME option: The default value is NO.
  - UPDATE\_MULT\_ROWS= LIBNAME option: This option is no longer supported.
  - USE\_ODBC\_CL= LIBNAME option: This option is no longer supported.
- In SAS 9.4M6, support was added for the CONOPTS= LIBNAME option and the SUB\_CHAR= LIBNAME option.

---

## SAS/ACCESS Interface to ODBC

In SAS 9.4, support was added for the DATETIME2= LIBNAME and data set options.

SAS 9.4M5 adds support for the DBNULLWHERE= LIBNAME option and the DBNULLWHERE= data set option.

In SAS Viya 3.3, integration was added between SAS/ACCESS Interface to ODBC and SAS Viya. This integration includes a data connector that enables you to load data to the CAS server.

In SAS Viya 3.4, the following changes or enhancements were made:

- Support for the 64-bit Windows platform for SAS Viya
  - Support for the following LIBNAME options was added:
    - DBSASLABEL=
    - PRESERVE\_USER=
    - WARN\_BIGINT=
  - The DBIDIRECTEXEC= system option is not supported for ODBC.
- In SAS Viya 3.5, the following changes or enhancements were added:
- Support for the Power Linux platform
  - CURSOR\_TYPE= LIBNAME option: There is no longer a default value for this option.
- In SAS 9.4M8, the following changes and enhancements were made:
- Support was added for the PRESERVE\_COMMENTS= LIBNAME option and ODBC\_SQL\_COMMENT macro variable. These options enable you to pass comments down to your data source in a query.
  - Support was added for ODBC for the DBIDIRECTEXEC system option.

---

## SAS/ACCESS Interface to OLE DB

In SAS 9.4, support for the BOOL\_VAL environment variable was added.

In SAS 9.4M3, these features have been added or enhanced:

- PRESERVE\_GUID= LIBNAME option
- COMPRESS function: This function can be passed to OLE DB only when you specify SQL\_FUNCTIONS=ALL.

In [SAS 9.4M4](#), support was added for the CHAR\_AS\_NCHAR= LIBNAME option.

---

## SAS/ACCESS Interface to Oracle

In [SAS 9.4](#), support was added for the OR\_IDENTITY\_COLS= data set option.

In [SAS 9.4M1](#), the following options were added or modified:

- DBCLIENT\_ENCODING\_FIXED= LIBNAME option
- DBSERVER\_ENCODING\_FIXED= LIBNAME option
- OR\_BINARY\_DOUBLE= LIBNAME option: The default value is YES.

[SAS 9.4M2](#) has these enhancements:

- BL\_API\_BULKLOAD= data set option: You can use this option to perform bulk loading using the Oracle Direct Path API.
- Starting with Oracle 12c, the default data type for character variables in SAS output has been changed to either VARCHAR2 or CLOB, depending on the length of the variable. When the NOTRANSCODE attribute is specified, the default data type can be either RAW or BLOB, depending on the length of the variable.

In [SAS 9.4M3](#), these options were added:

- POST\_STMT\_OPTS= [data set](#) and [LIBNAME](#) options
- POST\_TABLE\_OPTS= [data set](#) option
- PRE\_STMT\_OPTS= [data set](#) option
- PRE\_TABLE\_OPTS= [data set](#) option

In [SAS 9.4M4](#), support for the BL\_USE\_PIPE= data set option was added.

[SAS 9.4M5](#) adds support for these options:

- DBNULLWHERE= LIBNAME option and the DBNULLWHERE= data set option
- POST\_DML\_STMT\_OPTS= LIBNAME option and the POST\_DML\_STMT\_OPTS= data set option

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS Interface to Oracle and SAS Viya. This includes a data connector that enables you to load data to the CAS server.

Beginning in the April 2020 release of SAS/ACCESS for [SAS 9.4M6](#) and [SAS Viya 3.5](#), BL\_SQLLDR\_PATH= is no longer supported.

In [SAS 9.4M8](#), support was added for these changes and enhancements:

- support was added for the MAX\_STRING\_SIZE= LIBNAME option. This option enables you to explicitly specify the Oracle MAX\_STRING\_SIZE parameter value.

- support was added for the PRESERVE\_COMMENTS= LIBNAME option. This option enables you to pass comments down to your data source in a query.
- SAS/ACCESS Interface to Oracle on the z/OS platform is no longer supported. If you have an existing instance of SAS/ACCESS Interface to Oracle on z/OS and plan to upgrade to SAS 9.4M8 or later, then SAS/ACCESS recommends that you unconfigure and uninstall it. For more information, see [Unconfiguring and Uninstalling Retired Products](#).

---

# SAS/ACCESS Interface to PostgreSQL

In SAS 9.4, support was added for the PostgreSQL interface.

In [SAS 9.4M1](#), these features are new or modified:

- BL\_ESCAPE=, BL\_FORMAT=, BL\_NULL=, and BL\_QUOTE= data set options were added.
- PORT= connection option: The default value is 5432.

In [SAS 9.4M3](#), these features were added or enhanced:

- COMPRESS and COT functions: These functions can be passed automatically to PostgreSQL.
- These options were added:
  - POST\_STMT\_OPTS= [data set](#) and [LIBNAME](#) options
  - POST\_TABLE\_OPTS= data set option
  - PRE\_STMT\_OPTS= data set option
  - PRE\_TABLE\_OPTS= data set option

In [SAS 9.4M4](#), support for in-database processing of these Base SAS procedures was added: FREQ, MEANS, RANK, REPORT, SORT, SUMMARY, and TABULATE.

In [SAS 9.4M5](#), the default value for the DBINDEX= LIBNAME option and data set option is NO.

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS Interface to PostgreSQL and SAS Viya. This includes a data connector that enables you to load data to the CAS server in SAS Viya.

In [SAS Viya 3.4](#), the following changes or enhancements were made:

- Support for the 64-bit Windows platform on SAS Viya.
- Support was added for the following options:
  - COMPLETE= connection option
  - CURSOR\_TYPE= LIBNAME option and data set option
  - KEYSET\_SIZE= LIBNAME option and data set option
  - PROMPT= and NOPROMPT= connection options
  - QUALIFIER= LIBNAME option
  - SSLMODE= LIBNAME option

- The following options are not supported:
  - BL\_DEFAULT\_DIR= LIBNAME option

.....  
Note: The BL\_DEFAULT\_DIR= data set option is still supported.  
.....

- UPDATE\_MULT\_ROWS= LIBNAME option

In SAS 9.4M6, support was added for the CONOPTS= LIBNAME option and the SUB\_CHAR= LIBNAME option.

In SAS Viya 3.5, support was added for the following features and changes:

- Linux on the Power Architecture platform was added
- bulk unloading, which includes the BULKUNLOAD= data set option
- BL\_DELIMITER= data set option: The default value is the pipe symbol (|).
- INSERT\_SQL= LIBNAME option was added

In the March 2021 update for SAS/ACCESS for SAS 9.4M7 and SAS Viya 3.5, a clarification was added to the BL\_ESCAPE= data set option. For PostgreSQL, because BL\_ESCAPE= is applicable only to CSV files, you must also set BL\_FORMAT=CSV.

In the April 2022 update for SAS/ACCESS for SAS 9.4M7 and SAS Viya 3.5, the following changes have been made:

- The TRACE= and TRACEFILE= LIBNAME options are valid only for Windows platforms.
- The USE\_ODBC\_CL= LIBNAME option is not supported.

In the September 2022 update for SAS/ACCESS for SAS 9.4M7 and SAS Viya 3.5, a restriction was added for the SSLMODE= LIBNAME option. All values must be specified in lowercase for this option.

In SAS 9.4M8, the following enhancements and changes were made:

- Support has been added for in-database processing for PROC COPY.
- Support was added for the SAS\_POSTGRES\_UPDATE\_WARNING= environment variable to facilitate working with NLS data that would otherwise trigger errors in SAS.

---

## SAS/ACCESS Interface to SAP ASE

In SAS 9.4M3, these options were added:

- POST\_STMT\_OPTS= [data set](#) and [LIBNAME](#) options
- [POST\\_TABLE\\_OPTS=](#) data set option
- [PRE\\_STMT\\_OPTS=](#) data set option
- [PRE\\_TABLE\\_OPTS=](#) data set option

In **SAS 9.4M4**, the name SAS/ACCESS Interface to Sybase changed to SAS/ACCESS Interface to SAP ASE to match the changed SAP product name. In the LIBNAME statement, the engine name is now `sapase`, although the previous engine name, `sybase`, is still supported.

In SAS Viya 3.4, support was added for the SAP ASE interface in SAS Viya.

---

## SAS/ACCESS Interface to SAP HANA

In **SAS 9.4**, support was added for the SAP HANA interface.

In **SAS 9.4M2**, the `saphana` LIBNAME engine name was added.

In **SAS 9.4M3**, support was added for the following features

- views
- SQLGENERATION= LIBNAME and system options
- \*\* and COT functions: You can automatically pass down the SAS \*\* (POWER(base, exponent)) and COT functions to SAP HANA.
- PARMSTRING= and PARMDEFAULT= LIBNAME options and PARMSTRING= and PARMDEFAULT= data set options

In **SAS 9.4M4**, support for the READ\_LOCK\_TYPE= and UPDATE\_LOCK\_TYPE= LIBNAME options was added.

**SAS 9.4M5** adds support for the DBNULLWHERE= LIBNAME option and the DBNULLWHERE= data set option.

In **SAS Viya 3.3**, integration was added between SAS/ACCESS Interface to SAP HANA and SAS Viya. This includes a data connector that enables you to load data to the CAS server.

In **SAS 9.4M6**, the following changes and enhancements were made:

- Support was added for the CONOPTS= LIBNAME option.
- The MSCRYPTO value is no longer available for the SSLCRYPTOPROVIDER= LIBNAME option.

In the November 2020 update for **SAS 9.4M7**, support was added for the DBSASLABEL= LIBNAME option and the DBMS value in the DBSASLABEL= data set option.

---

## SAS/ACCESS Interface to SAP IQ

In **SAS 9.4M3**, these options were added:

- POST\_STMT\_OPTS= [data set](#) and [LIBNAME](#) options
- POST\_TABLE\_OPTS= [data set](#) option

- [PRE\\_STMT\\_OPTS=](#) data set option
- [PRE\\_TABLE\\_OPTS=](#) data set option

In SAS 9.4M4, the name SAS/ACCESS Interface to Sybase IQ changed to SAS/ACCESS Interface to SAP IQ to match the changed SAP product name. In the LIBNAME statement, the engine name is now `sapiq`, although the previous engine name, `sybaseiq`, is still supported.

In SAS 9.4M6, support was added for the CONOPTS= LIBNAME option.

In SAS 9.4M7, the output from PROC CONTENTS and PROC DATASETS now includes the SAP IQ client version and server version. The output for PROC DATASETS also includes the LIBNAME engine.

In SAS 9.4M8, the following changes and enhancements were added:

- Support was added for the DBCLIENT\_MAX\_BYTES= LIBNAME option.

---

## SAS/ACCESS Interface to Snowflake

In the August 2019 release of SAS 9.4M6 and SAS Viya 3.4, support was added for the Snowflake interface.

In SAS Viya 3.5, support was added for the 64-bit Microsoft Windows platform.

In the May 2020 update of SAS/ACCESS for SAS 9.4M6 and SAS Viya 3.5, the following changes and enhancements were made:

- When the DBIDIRECTEXEC= system option is enabled, you can update and insert content into a table that has a primary key.
- You can now emulate a positional cursor in Snowflake data.
- Both of these are now required: The SCHEMA= LIBNAME option and data set option.
- The default value for the DBCLIENT\_MAX\_BYTES= LIBNAME option now matches the maximum number of bytes per single character of the SAS session encoding.

In the July 2021 update for SAS 9.4M7 and SAS Viya 3.5, support was added for the DBMSTEMP= LIBNAME option.

In SAS 9.4M8, support for the [BL\\_FORMAT\\_OPT=](#) data set option was added.

---

## SAS/ACCESS Interface to Spark

In SAS 9.4M7, support was added for the Spark interface on SAS 9.4. Also in this release, SAS/ACCESS Interface to Spark is no longer available on SAS Viya 3.4 or on SAS Viya 3.5.

In SAS 9.4M8, the following changes and enhancements were made:

- ability to access data in Databricks. To connect to Databricks, you must obtain, install, and configure the Databricks JDBC driver. For more information, see [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#).
- support for JDBC drivers other than the Apache Hive open source driver. For more information, see [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#).
- ability to specify additional resources to add to the JDBC class path. Support has been added for the CLASSPATH= LIBNAME option.
- support for enhanced transaction logging. This is controlled with the DRIVER\_TRACE= and related options.
- ability to override the default read buffer size. Support has been added for the READBUFF= LIBNAME and data set options.

---

## SAS/ACCESS Interface to Teradata

In [SAS 9.4](#), these features are new or enhanced:

- support for the DBCLIENT\_MAX\_BYTES= LIBNAME option
- support for the SAS\_DBMS\_AUTOMETADATA= LIBNAME option and the SAS\_DBMS\_AUTOMETADATA= environment variable
- TPT\_MAX\_SESSIONS= data set option: This option has a new default value.
- TPT messages in PROC SQL log output
- DATEPART function: This function is now passed down by default.

In [SAS 9.4M3](#), these features are new or enhanced:

- SAS/ACCESS supports object names that contain up to 32 characters for users who use Teradata 14.10 or later.
- Support was added for the Teradata Wallet security feature.
- The POST\_STMT\_OPTS= [data set](#) and [LIBNAME](#) options are available.
- The [POST\\_TABLE\\_OPTS=](#) data set option is available.
- The [PRE\\_STMT\\_OPTS=](#) data set option is available.
- The [PRE\\_TABLE\\_OPTS=](#) data set option is available.
- The [TPT\\_MIN\\_SESSIONS=](#) LIBNAME option is available.
- Function updates: The SUM4 function was removed from the list of functions that can be passed to Teradata. The TRIM function can be passed automatically and does not require SQL\_FUNCTIONS=ALL. The LENGTH function does require SQL\_FUNCTIONS=ALL.

In [SAS 9.4M4](#), support has been added for single sign-on with Kerberos authentication.

In [SAS Viya 3.3](#), integration was added between SAS/ACCESS Interface to Teradata and SAS Viya. This includes a data connector that enables you to load

data onto the CAS server. You can move data in parallel if you also license SAS In-Database Technologies.

The [SAS Viya 3.5](#) release includes the following changes or enhancements:

- Support for the Teradata NUMBER data type was added.
- There is no default value for the ERRLIMIT= LIBNAME option.

In [SAS 9.4M8](#), support was added for the following changes and enhancements:

- Support has been added for the [OVERRIDE\\_RESP\\_LEN= LIBNAME](#) option. This option enables you to specify a response buffer length that is larger than the default length of 1,048,500 bytes when you are not using the TPT API.
- Support has been added for the [TD\\_1MB\\_ROW environment variable](#). This environment variable works with the TPT API and enables SAS/ACCESS Interface to Teradata to retrieve response rows up to 1MB.
- SAS/ACCESS Interface to Teradata on the z/OS platform is no longer available. If you have an instance of SAS/ACCESS Interface to Teradata on z/OS and plan to upgrade to SAS 9.4M8 or later, SAS recommends that you first unconfigure and uninstall it. For more information, see [Unconfiguring and Uninstalling Retired Products](#).

---

## SAS/ACCESS Interface to Vertica

[SAS 9.4](#) includes support for the Vertica interface.

In [SAS 9.4M3](#), these options were added:

- [POST\\_STMT\\_OPTS= data set](#) and [LIBNAME](#) options
- [POST\\_TABLE\\_OPTS= data set option](#)
- [PRE\\_STMT\\_OPTS= data set option](#)
- [PRE\\_TABLE\\_OPTS= data set option](#)

In [SAS 9.4M4](#), support for in-database processing of these Base SAS procedures was added: FREQ, MEANS, RANK, REPORT, SORT, SUMMARY, and TABULATE.

In [SAS Viya 3.4](#), support was added for the SAS Data Connector to Vertica. The data connector enables you to transfer large amounts of data between your Vertica database and the CAS server.

In [SAS 9.4M6](#), the following changes and enhancements were made:

- Support was added for the CONOPTS= LIBNAME option
- There is no longer support for the 32-bit Solaris platform.

---

## SAS/ACCESS Interface to Yellowbrick

Support for the Yellowbrick interface was added in [SAS 9.4M7](#). This includes support for bulk loading, bulk unloading, and in-database processing of summary procedures, such as PROC MEANS.

In the April 2022 update for SAS/ACCESS for [SAS 9.4M7](#) and [SAS Viya 3.5](#), the following changes have been made:

- The TRACE= and TRACEFILE= LIBNAME options are valid only for Windows platforms.
- The USE\_ODBC\_CL= LIBNAME option is not supported.

In the September 2022 update for SAS/ACCESS for [SAS 9.4M7](#) and [SAS Viya 3.5](#), a restriction was added for the SSLMODE= LIBNAME option. All values must be specified in lowercase for this option.

In [SAS 9.4M8](#), information was added about how to work with long character values in Yellowbrick.

---

## Documentation Enhancements

This document includes these new and enhanced items.

- In [SAS 9.4M3](#), these additions or updates were made throughout the documentation:
  - Documentation of the HDMD Procedure was moved to *Base SAS Procedures Guide*.
  - The SQL pass-through function MOD might yield different results from SAS if non-integer arguments are specified. See the documentation for your DBMS for details.
- In [SAS Viya 3.3](#), these additions or updates were made in the documentation:
  - In [Chapter 11, “SAS/ACCESS Features by Host,” on page 95](#), applicable interfaces have information about supported features and platforms for SAS Viya.
  - The CONNECTION= LIBNAME option was updated to state that only the GLOBAL and SHARED values are passed to the DBMS in a PROC SQL CONNECT statement.
- Aster: In [SAS 9.4M4](#), the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.
- DB2 under UNIX and PC Hosts: In [SAS 9.4M4](#), the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.

- DB2 under z/OS:
  - In [SAS 9.4M3](#), the list of default SAS formats for various data types has been updated.
  - In [SAS 9.4M4](#), the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.
- Greenplum:
  - In [SAS 9.4M4](#), the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.
  - In [SAS Viya 3.4](#), the following aliases were added to the associated option:
    - DSN= connection option: DS= and DATASRC= aliases were added.
    - IGNORE\_READONLY\_COLUMNS= LIBNAME option and IGNORE\_READONLY\_COLUMNS data set option: IGNORE\_READONLY= alias was added.
    - KEYSET\_SIZE= LIBNAME option and KEYSET\_SIZE= data set option: KEYSET= alias was added.
    - PORT= connection option: SERVICE= and SERVICE\_NAMES= aliases were added.
    - QUERY\_TIMEOUT= LIBNAME option and QUERY\_TIMEOUT= data set option: TIMEOUT= alias was added.
    - READ\_ISOLATION\_LEVEL= LIBNAME option and READ\_ISOLATION\_LEVEL= data set option: RIL= alias was added.
    - SCHEMA= LIBNAME option and SCHEMA= data set option: OWNER= alias was added.
    - SERVER= connection option: HOST= alias was added.
    - STRINGDATES= LIBNAME option: STRDATES= alias was added.
    - USER= connection option: UID= alias was added.
  - In [SAS 9.4M6](#), information about locking in the Greenplum interface was added.
- Hadoop:

In SAS Viya 3.4 and in SAS 9.4M5, the following documentation changes were made:

  - The LIBNAME option CONFIG= is no longer documented. CONFIGDIR= is an alias for the CONFIG= LIBNAME option.
  - The deprecated LIBNAME option SUBPROTOCOL= has been removed from the documentation.
- HAWQ:
  - In [SAS 9.4M4](#), the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.
  - In [SAS Viya 3.4](#), these documentation changes were made:
    - DSN= connection option: DS= and DATASRC= aliases were documented.
    - IGNORE\_READONLY\_COLUMNS= LIBNAME option and IGNORE\_READONLY\_COLUMNS data set option: IGNORE\_READONLY= alias was documented.

- KEYSET\_SIZE= LIBNAME option and KEYSET\_SIZE= data set option: The KEYSET= alias was documented.
- PORT= connection option: SERVICE= and SERVICE\_NAMES= aliases were documented.
- QUERY\_TIMEOUT= LIBNAME option and QUERY\_TIMEOUT= data set option: TIMEOUT= alias was documented.
- READ\_ISOLATION\_LEVEL= data set option: RIL= alias was documented.
- SCHEMA= LIBNAME option and SCHEMA= data set option: OWNER= alias was documented.
- SERVER= connection option: HOST= alias was documented.
- STRINGDATES= LIBNAME option: STRDATES= alias was documented.
- USER= connection option: UID= alias was documented.
- In SAS 9.4M6, documentation about locking in the HAWQ interface was added.
- Impala:
  - In SAS 9.4M4, support for new configuration-specific details were added to the bulk-load section.
- JDBC: In SAS 9.4M6, the following options were removed from the list of options that are supported for JDBC:
  - DELETE\_MULT\_ROWS= LIBNAME option
  - DIRECT\_EXE= LIBNAME option
  - ERRLIMIT= data set option
  - IGNORE\_READ\_ONLY\_COLUMNS= LIBNAME option
  - INSERT\_SQL= LIBNAME option
  - SASDATEFMT= data set option
- Microsoft SQL Server:
  - In SAS 9.4M4, the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.
  - In SAS 9.4M5, support is no longer indicated in the documentation for Microsoft SQL Server for the DBIRECTEXEC system option.
- MySQL: In the April 2016 release for SAS/ACCESS on SAS 9.4M3, the list of LIBNAME options that are supported in the CONNECT statement for the SQL procedure are identified.
- Netezza: SAS 9.4M5 documentation no longer indicates that the DELETE\_MULT\_ROWS= or UPDATE\_MULT\_ROWS= LIBNAME options are supported.
- ODBC: In SAS 9.4M4, the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.
- Oracle:
  - In the April 2016 release for SAS/ACCESS on SAS 9.4M3, the list of LIBNAME options that are supported in the CONNECT statement for the SQL procedure are identified.

- In SAS 9.4M6, information about how to specify user names for bulk loading was added.
  - PostgreSQL:
    - In SAS 9.4M3, a new section, “Working with Long Character Values in PostgreSQL,” was added.
    - In the April 2016 release for SAS/ACCESS on SAS 9.4M3, the list of LIBNAME options that are supported in the CONNECT statement for the SQL procedure are identified.
  - SAP ASE: In SAS Viya 3.3, the list of LIBNAME options that are supported in the CONNECT statement for the SQL procedure are identified.
  - SAP HANA:
    - In SAS 9.4M2, the documentation explains how to work with SAP HANA geospatial data.
    - In SAS 9.4M3, a new section, “SAP HANA Schema Flexibility,” was added.
    - In SAS 9.4M4, the list of LIBNAME options now identifies the options that are valid in the CONNECT statement for the SQL procedure.
  - SAP IQ:
    - In the April 2016 release for SAS/ACCESS, the list of LIBNAME options that are supported in the CONNECT statement for the SQL procedure are identified.
    - In SAS Viya 3.3, aliases were added for these options
      - connection options: DATABASE=, DSN=, HOST=, PASSWORD=, PORT=, SERVER=, and USER=.
      - LIBNAME options: CURSOR\_TYPE=, IGNORE\_READ\_ONLY\_COLUMNS=, PRESERVE\_COL\_NAMES=, QUERY\_TIMEOUT=, READ\_ISOLATION\_LEVEL=, SCHEMA=, STRINGDATES=, and UPDATE\_ISOLATION\_LEVEL=.
      - data set options: BULKLOAD=, CURSOR\_TYPE=, IGNORE\_READ\_ONLY\_COLUMNS=, PRESERVE\_COL\_NAMES=, QUERY\_TIMEOUT=, READ\_ISOLATION\_LEVEL=, SCHEMA=, and UPDATE\_ISOLATION\_LEVEL=.
  - Teradata:
    - In the April 2016 release for SAS/ACCESS on SAS 9.4M3, the list of LIBNAME options that are supported in the CONNECT statement for the SQL procedure are identified.
    - In SAS Viya 3.5, information was added about how to add comments to SQL code when using PROC SQL. For more information, see “[Including Comments in SQL Code](#)” on page 1333.

**PART 1**

# Concepts

<i>Chapter 1</i>	
<i>Overview of SAS/ACCESS Interface to Relational Databases</i>	3
<i>Chapter 2</i>	
<i>Using SAS/ACCESS with SAS Viya</i>	13
<i>Chapter 3</i>	
<i>SAS Names and Support for DBMS Names</i>	21
<i>Chapter 4</i>	
<i>Data Integrity and Security</i>	37
<i>Chapter 5</i>	
<i>Performance Considerations</i>	47
<i>Chapter 6</i>	
<i>Optimizing Your SQL Usage</i>	55
<i>Chapter 7</i>	
<i>Threaded Reads</i>	69
<i>Chapter 8</i>	
<i>National Language Support</i>	79
<i>Chapter 9</i>	
<i>How SAS/ACCESS Works</i>	81
<i>Chapter 10</i>	
<i>In-Database Processing with SAS/ACCESS</i>	89



# Overview of SAS/ACCESS Interface to Relational Databases

---

<i>About SAS/ACCESS Documentation</i> .....	3
<i>Sample Code</i> .....	4
<i>Methods for Accessing Relational Database Data</i> .....	4
<i>Selecting a SAS/ACCESS Method</i> .....	5
Methods for Accessing DBMS Tables and Views .....	5
SAS/ACCESS LIBNAME Statement Advantages .....	5
SQL Pass-Through Facility Advantages .....	6
SAS/ACCESS Features for Common Tasks .....	6
<i>SAS Views of DBMS Data</i> .....	7
<i>Choosing Your Degree of Numeric Precision</i> .....	8
Factors That Can Cause Calculation Differences .....	8
Examples of Problems That Result in Numeric Imprecision .....	9
Your Options When Choosing Your Needed Degree of Precision .....	10
References .....	11
National Language Support .....	11

---

## About SAS/ACCESS Documentation

This documentation provides conceptual, reference, and usage information for SAS/ACCESS software for data sources for relational database management systems (DBMSs), data warehouse appliances, and distributed systems.

The availability and behavior of SAS/ACCESS features vary from one interface to another. Therefore, consult the information in both the general and DBMS-specific sections of this document when working with your particular SAS/ACCESS interface. Also, refer to the SAS system requirements and configuration guide documents that are available at <http://support.sas.com>.

This document assumes that you are an applications programmer or end user with these skills.

- You are familiar with the basics of their DBMS or data warehouse appliance and its Structured Query Language (SQL).
- If you use Hadoop, you are familiar with Hadoop and the Hadoop Distributed File System (HDFS). You have knowledge of fundamental Hadoop library and processing using HDFS. You are also familiar with the basics of Hadoop, Hive, and HiveQL. For details, see the Apache Hadoop and Hive websites at <http://hadoop.apache.org> and <https://cwiki.apache.org/confluence/display/Hive>.
- You know how to use your operating environment.
- You can use basic SAS commands and statements.

If you are a database administrator, you might also want to read this document to understand how to implement and administer specific interfaces.

---

## Sample Code

Sample code is available for SAS/ACCESS. This sample code contains examples that use the LIBNAME statement to associate librefs with DBMS objects such as tables.

You can find current samples at the [SAS/ACCESS GitHub location](#).

---

## Methods for Accessing Relational Database Data

SAS/ACCESS Interface to Relational Databases is a family of interfaces—each licensed separately—with which you can interact with data in other vendor databases from within SAS. SAS/ACCESS provides these methods for accessing relational DBMS data.

- To assign SAS librefs to DBMS objects such as schemas and databases, you can use the [LIBNAME statement](#). After you associate a database with a libref, you can use a SAS two-level name to specify any table or view in the database. You can then work with the table or view as you would with a SAS data set.
- To interact with a data source using its native SQL syntax without leaving your SAS session, you can use the [SQL pass-through facility](#). SQL statements are passed directly to the data source for processing.
- For indirect access to DBMS data, you can use [ACCESS](#) and [DBLOAD](#) procedures. Although SAS still supports these legacy procedures for specific DBMSs and environments, they are no longer the recommended method for accessing DBMS data.

Not all SAS/ACCESS interfaces support all of these features. To determine which features are available in your environment, see [SAS/ACCESS Features by Host on page 96](#).

---

# Selecting a SAS/ACCESS Method

---

## Methods for Accessing DBMS Tables and Views

You can often complete a task in SAS/ACCESS in several ways. For example, you can access DBMS tables and views by using the [LIBNAME statement](#) or the [SQL pass-through facility](#). Before processing complex or data-intensive operations, you might want to test different methods first to determine the most efficient one for your particular task.

---

## SAS/ACCESS LIBNAME Statement Advantages

You should use the SAS/ACCESS LIBNAME statement for the fastest and most direct method of accessing your DBMS data except when you need to use SQL that is not ANSI-standard. ANSI-standard SQL is required when you use the SAS/ACCESS library engine in the SQL procedure. However, the SQL pass-through facility accepts all SQL extensions that your DBMS provides.

Here are the advantages of using the SAS/ACCESS LIBNAME statement.

- Significantly fewer lines of SAS code are required to perform operations on your DBMS. For example, a single LIBNAME statement establishes a connection to your DBMS, lets you specify how data is processed, and lets you easily view your DBMS tables in SAS.
- You do not need to know the SQL language of your DBMS to access and manipulate data on your DBMS. You can use such SAS procedures as PROC SQL or DATA step programming on any libref that references DBMS data. You can read, insert, update, delete, and append data. You can also create and drop DBMS tables by using SAS syntax.
- The LIBNAME statement gives you more control over DBMS operations such as locking, spooling, and data type conversion through the use of LIBNAME and data set options.
- The engine can optimize processing of joins and WHERE clauses by passing them directly to the DBMS, which takes advantage of the indexing and other processing capabilities of your DBMS. For more information, see [Optimizing Your SQL Usage on page 55](#).
- The engine can pass some functions directly to the DBMS for processing.

## SQL Pass-Through Facility Advantages

Here are the advantages of using the SQL pass-through facility.

- You can use SQL pass-through facility statements so that the DBMS can optimize queries, particularly when you join tables. The DBMS optimizer can take advantage of indexes on DBMS columns to process a query more quickly and efficiently.
- SQL pass-through facility statements let the DBMS optimize queries when queries have summary functions. Summary functions include AVG, COUNT, GROUP BY clauses, or columns that are created by expressions, such as those that use the COMPUTED function. The DBMS optimizer can use indexes on DBMS columns to process queries more rapidly.
- On some DBMSs, you can use SQL pass-through facility statements with SAS/AF applications to handle transaction processing of DBMS data. Using a SAS/AF application gives you complete control of COMMIT and ROLLBACK transactions. SQL pass-through facility statements give you better access to DBMS return codes.
- The SQL pass-through facility accepts all extensions to ANSI SQL that your DBMS provides.

## SAS/ACCESS Features for Common Tasks

Here is a list of tasks and the features that you can use to accomplish them.

*Table 1.1 SAS/ACCESS Features for Common Tasks*

Task	SAS/ACCESS Features
Read DBMS tables or views	LIBNAME statement <sup>1</sup> SQL pass-through facility
Create DBMS objects, such as tables	LIBNAME statement <sup>1</sup> SQL pass-through facility EXECUTE statement
Update, delete, or insert rows into DBMS tables	LIBNAME statement <sup>1</sup> SQL pass-through facility EXECUTE statement
Append data to DBMS tables	LIBNAME statement and APPEND procedure <sup>1</sup> SQL pass-through facility EXECUTE statement

Task	SAS/ACCESS Features
	SQL pass-through facility INSERT statement
List DBMS tables	LIBNAME statement and SAS Explorer window <sup>1</sup>
	LIBNAME statement and DATASETS procedure <sup>1</sup>
	LIBNAME statement and CONTENTS procedure <sup>1</sup>
	LIBNAME statement and SQL procedure dictionary tables <sup>1</sup>
Delete DBMS tables or views	LIBNAME statement and SQL procedure DROP TABLE statement <sup>1</sup>
	LIBNAME statement and DATASETS procedure DELETE statement <sup>1</sup>
	SQL pass-through facility EXECUTE statement

<sup>1</sup> LIBNAME statement refers to the SAS/ACCESS LIBNAME statement.

## SAS Views of DBMS Data

SAS/ACCESS lets you create a SAS view of data that exists in a DBMS. A SAS *data view* specifies a virtual data set that is named and stored for later use. A view contains no data but instead describes data that is stored elsewhere. Here are the types of SAS data views.

**DATA step views**  
stored, compiled DATA step programs.

**SQL views**  
are stored query expressions that read data values from their underlying files, which can include SAS data sets, SAS/ACCESS views, DATA step views, other SQL views, or relational database data.

You can use all types of views as inputs into DATA steps and procedures. You can specify views in queries as if they were tables. A view derives its data from the tables or views that are listed in its FROM clause. The data accessed by a view is a subset or superset of the data in its underlying table(s) or view(s).

You can use SQL views and SAS/ACCESS views to update their underlying data if one of the following is true:

- The view is based on only one DBMS table.
- The view is based on a DBMS view that is based on only one DBMS table, and the view has no calculated fields.

You cannot use DATA step views to update the underlying data; you can use them only to read the data.

Your options for creating a SAS view of DBMS data are determined by the SAS/ACCESS feature that you are using to access the DBMS data. This table lists the recommended methods for creating SAS views.

**Table 1.2** Creating SAS Views

Feature for Accessing DBMS Data	SAS View Technology to Use
SAS/ACCESS LIBNAME statement	SQL view or DATA step view of the DBMS table
SQL pass-through facility	SQL view with CONNECTION TO component

---

## Choosing Your Degree of Numeric Precision

---

### Factors That Can Cause Calculation Differences

Different factors affect numeric precision. This issue is common for many people, including SAS users. Though computers and software can help, you are limited in how precisely you can calculate, compare, and represent data. Therefore, only those people who generate and use data can determine the exact degree of precision that meets their enterprise needs.

As you decide the degree of precision that you want, you need to consider that these system factors can cause calculation differences:

- hardware limitations
- differences among operating systems
- different software or different versions of the same software
- different DBMSs

These factors can also cause differences:

- the use of finite number sets to represent infinite real numbers
- how numbers are stored, because storage sizes can vary

You also need to consider how conversions are performed on, between, or across any of these system or calculation factors.

---

# Examples of Problems That Result in Numeric Imprecision

---

## Overview

Depending on the degree of precision that you want, calculating the value of  $r$  can result in a tiny residual in a floating-point unit. When you compare the value of  $r$  to 0.0, you might find that  $r \neq 0.0$ . The numbers are very close but not equal. This type of discrepancy in results can stem from problems in representing, rounding, displaying, and selectively extracting data.

---

## Representing Data

Some numbers can be represented exactly, but others cannot. As shown in this example, the number 10.25, which terminates in binary, can be represented exactly.

```
data x;
  x=10.25;
  put x hex16.;
run;
```

The output from this DATA step is an exact number: 4024800000000000. However, the number 10.1 cannot be represented exactly, as this example shows.

```
data x;
  x=10.1;
  put x hex16.;
run;
```

The output from this DATA step is an inexact number: 4024333333333333.

---

## Rounding Data

As this example shows, rounding errors can result from platform-specific differences. No solution exists for such situations.

```
data x;
  x=10.1;
  put x hex16.;
  y=100000;
  newx=(x+y)-y;
  put newx hex16.;
run;
```

In Linux environments, the output from this DATA step is 4024333333333333 (8/10-byte hardware double).

## Displaying Data

For certain numbers such as x.5, the precision of displayed data depends on whether you round up or down. Low-precision formatting (rounding down) can produce different results on different platforms. In this example, the same high-precision (rounding up) result occurs for X=8.3, X=8.5, or X=hex16. However, a different result occurs for X=8.1 because this number does not yield the same level of precision.

```
data;
x=input('C047DFFFFFFFFF', hex16.);
put x= 8.1 x= 8.3 x= 8.5 x= hex16.;
run;
```

Here is the output under Linux (high-precision formatting).

```
x=-47.8
x=-47.750 x=-47.7500
x=C047DFFFFFFFFF
```

To fix the problem that this example illustrates, you must select a number that yields the next precision level—in this case, 8.2.

## Your Options When Choosing Your Needed Degree of Precision

After you determine the degree of precision that your enterprise needs, you can refine your software. You can use macros, sensitivity analyses, or fuzzy comparisons such as extractions or filters to extract data from databases or from different versions of SAS. For example, you can use this EQFUZZ macro.

```
*****
*/
/* This macro defines an EQFUZZ operator. The subsequent DATA step
shows      */
/* how to use this operator to test for equality within a certain
tolerance. */
/
*****
*/
%macro eqfuzz(var1, var2, fuzz=1e-12);
abs((&var1 - &var2) / &var1) < &fuzz
%mend;

data _null_;
x=0;
y=1;
do i=1 to 10;
```

```

x+0.1;
end;
if x=y then put 'x exactly equal to y';
else if %eqfuzz(x,y) then put 'x close to y';
else put 'x nowhere close to y';
run;

```

When you read numbers in from an external DBMS that supports precision beyond 15 digits, you can lose that precision. You cannot do anything about this for existing databases. However, when you design new databases, you can set constraints to limit precision to about 15 digits. Alternatively, you can select a numeric DBMS data type to match the numeric SAS data type. For example, select the DOUBLE type in Hadoop (precise up to 15 digits) or the INT type instead of the BIGINT type (precise up to 38 digits).

When you read numbers in from an external DBMS for noncomputational purposes, use the DBSASTYPE= data set option, as shown in this example.

```

libname x hadoop user=myusr1 password=mypwd1 database=db;
data sasdata;
  set ora.catalina2(dbsastype= (c1='char(20)' ));
run;

```

This option retrieves numbers as character strings and preserves precision beyond 15 digits. For details, see the [DBSASTYPE= data set option on page 529](#).

## References

See these resources for more detail about numeric precision, including variables that can affect precision.

- [“Numeric Precision” in SAS Programmer’s Guide: Essentials](#)
- The Aggregate. 2008. “Numerical Precision, Accuracy, and Range,” Aggregate.org: *Unbridled Computing*. Lexington, KY: University of Kentucky. Available at <http://aggregate.org/NPAR>. Accessed on July 20, 2017.
- IEEE. 2008. “IEEE 754: Standard for Binary Floating-Point Arithmetic.” Available at <http://grouper.ieee.org/groups/754/index.html>. This standard specifies 32-bit and 64-bit floating-point representations and computational results. Accessed on July 20, 2017.
- SAS Institute Inc. 2007. “Dealing with Numeric Representation Error in SAS Applications,” technical paper. Cary, NC: SAS Institute Inc. Available at <http://support.sas.com/resources/papers/dealing-with-numeric-representation-error-in-sas-applications.pdf>. Accessed on July 20, 2017.

## National Language Support

SAS/ACCESS provides National Language Support (NLS) in a variety of ways.

These LIBNAME or SQL pass-through options allow for byte and character conversion and length calculation.

- ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS=

- ADJUST\_NCHAR\_COLUMN\_LENGTHS=
- DB\_LENGTH\_SEMANTICS\_BYTE=
- DBCLIENT\_MAX\_BYTES=
- DBSERVER\_MAX\_BYTES=

These data types allow for more flexible adjustment of column lengths.

- BLOB
- CHAR
- CLOB
- DBCLOB
- NCHAR
- VARCHAR

For more NLS information, see these resources.

- For NLS limitations that are specific to Hive, see “[Naming Conventions for SAS and Hive](#)” on page 912.
- For additional NLS considerations, see the technical paper, “[Processing Multilingual Data with the SAS 9.2 Unicode Server](#).” This paper is available at this URL:  
<http://support.sas.com/resources/papers/92unicodesrvr.pdf>.
- For more comprehensive NLS information, see [\*SAS National Language Support \(NLS\): Reference Guide\*](#).

# Using SAS/ACCESS with SAS Viya

---

<i>How SAS/ACCESS Works with the CAS Server</i> .....	13
<b>SAS/ACCESS Usage with SAS Viya</b> .....	15
Overview of Data Flow for SAS Viya .....	15
Tools That Are Available .....	16
Suggested Workflow for Data Sources with Data Connectors .....	16
Suggested Workflow for Data Sources without Data Connectors .....	17

---

## How SAS/ACCESS Works with the CAS Server

When you use SAS Viya, most of your analysis takes place using the CAS server. CAS holds your data in memory for fast access, and the CAS server enables distributed processing of large amounts of data. Use your data connector to load your data into a caslib that you add using a [CASLIB statement](#). If you have a data connect accelerator for your interface, use it to load data in parallel into CAS. For more information, see “[Working with SAS Data Connectors](#)” in [SAS Cloud Analytic Services: User’s Guide](#).

These SAS/ACCESS interfaces have corresponding data connectors.

Amazon Redshift	ODBC
DB2 under UNIX hosts	Oracle
Google BigQuery	PostgreSQL
Hadoop *	SAP HANA
Impala	Snowflake
JDBC	Teradata *
Microsoft SQL Server	Vertica

## MySQL

- \* It is possible to transfer data in parallel if you have separately licensed SAS In-Database Technologies for this interface.

*Additional interfaces:* In addition to the relational DBMS interfaces in this book, the following SAS/ACCESS interfaces have corresponding data connectors:

MongoDB    Salesforce  
PC Files

For more information, see “[Data Connectors](#)” in *SAS Cloud Analytic Services: User’s Guide*.

In addition, some SAS/ACCESS interfaces can be included with SAS Viya but do not have corresponding data connectors, such as Greenplum, HAWQ, Netezza, and SAP ASE.

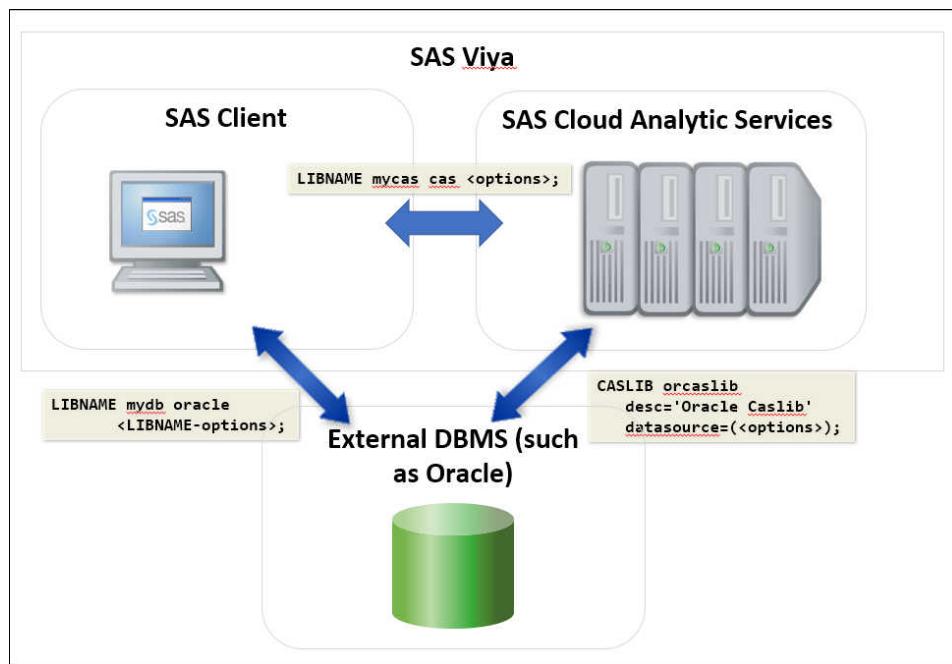
Your traditional SAS/ACCESS interface enables you to establish a connection, via a libref that you specify in a [LIBNAME](#) statement, between the SAS client and your external data source. After you specify this connection to your data source, you can send SQL commands by using the SQL procedure to manipulate the data in your external data source.

To access results that you generate in CAS, use a CAS LIBNAME statement to specify a libref between the SAS client and the CAS server. For more information, see “[CAS LIBNAME Engine Overview](#)” in *SAS Cloud Analytic Services: User’s Guide*.

# SAS/ACCESS Usage with SAS Viya

## Overview of Data Flow for SAS Viya

*Figure 2.1 Statements That Move Data in SAS Viya*



The preceding figure shows the SAS statements that you use to move your data between SAS Viya components and your external data source.

If you have a data connector, use a CASLIB statement to open a connection between your data source, such as an Oracle database, and the CAS server. You then use the CASUTIL or CAS procedures to load your data into CAS.

To open a connection between the SAS client and your database, use a LIBNAME statement for your database engine. Use this statement if there is not a data connector for your interface or if you do not need to use the CAS server to analyze your data. You can then work with your data using a DATA step, a call to PROC SQL, or other SAS statements.

To open a connection between the SAS client and the CAS server, use a LIBNAME statement for the CAS engine.

---

## Tools That Are Available

### SAS/ACCESS interface

enables you to connect your external data source with the local SAS client. Use the [LIBNAME statement](#) to connect to a data source location.

### CAS LIBNAME engine

enables you to connect the local SAS client with your CAS session so that you can access data in CAS tables. Use the [CAS LIBNAME statement](#) to connect to a CAS session.

### data connector or data connect accelerator

enables you to specify options that are used to connect your external data source with the CAS server. For more information, see “[Quick Reference for Data Connector Syntax](#)” in *SAS Cloud Analytic Services: User’s Guide*.

By using the [LOAD statement](#) in PROC CASUTIL, you use a data connector to load data into a caslib on the CAS server.

By using the [SAVE statement](#) in PROC CASUTIL, you use a data connector to save CAS data to a table in your external data source.

### SQL procedure

enables you to manipulate data in your external data source. Use PROC SQL to perform joins or filter your data before loading it into a caslib. For more information, see the information about using the SQL pass-through facility for your data source or *SAS SQL Procedure User’s Guide*.

---

## Suggested Workflow for Data Sources with Data Connectors

Here are the general steps to manage and manipulate your data from an external data source that has a corresponding data connector.

- 1 Perform any necessary data preparation before loading data into CAS. For example, if you want to load the join result of two Hive tables into CAS, create a Hive table or view that is the result of the join. You can then load the resulting table or view into CAS.  
Perform the data preparation by interacting directly with your data source or by using the SQL procedure in SAS.
- 2 Load the data into CAS using a data connector or data connect accelerator. For more information, see “[Quick Reference for Data Connector Syntax](#)” in *SAS Cloud Analytic Services: User’s Guide*.
- 3 Perform analysis on your data. You can use the DATA step, procedures that work with CAS, and CAS actions on your data. Save results to CAS tables in a caslib. For more information, see [SAS Viya Quick Start](#) and *SAS Cloud Analytic Services: Fundamentals*.

- 4 (Optional) Use the data connector, via PROC CASUTIL, to save results back to your external data source. As shown in the code below, you can save a result table, Casresults, using a SAVE statement that saves data in the Hadoop data source in a table called Myresults.

```

/* Add a caslib for loading and analyzing Hadoop data. */
caslib casref datasource=(srctype='hadoop',
                     dataTransferMode='parallel',
                     server='HiveServer2',
                     username='user1'
                     hadoopJarPath="//root/myJars",
                     hadoopConfigDir("//sasroot/myConfig"));

/* Specify a libref to access a Hadoop
   database. */
/* Although it is not shown here, you can use PROC SQL to access data
   in */
/* libref h to merge or filter data prior to reading it into
   CAS. */
libname h hadoop user=user1 pwd=mypwd1 server='HiveServer2'
schema=statsdiv;

/* Specify a libref to connect the SAS client with a CAS
   session. */
/* By default, the connection goes to the active
   caslib. */
/* You can run procedures that are not CAS procedures by
   accessing */
/* CAS data using the c
   libref. */
libname c cas;

/* Load data into CAS. */
proc casutil incaslib="casref";
  load casdata="myhivedata" casout="mycasdata";
quit;

/* Additional code to run analyses goes here */

/* Write results table back out to Hadoop */
proc casutil incaslib="casref";
  save casdata="casresults" casout="Myresults";
quit;

```

---

## Suggested Workflow for Data Sources without Data Connectors

If you have a SAS/ACCESS interface that is supported in SAS Viya but that does not have a corresponding data connector, you can still load your data onto the CAS server. Here are the interfaces that fall into this category:

- Greenplum
- HAWQ

- Netezza
- SAP ASE

The R/3 interface is also supported in SAS Viya, although it does not have a corresponding data connector. For more information about the R/3 interface, see [SAS/ACCESS Interface to R/3: User's Guide](#).

Here are the general steps to manage and manipulate your data in SAS Viya without a corresponding data connector.

- 1 Perform any necessary data preparation before loading data into the SAS client. For example, if you want to load the join result of two Hive tables into CAS, create a Hive table that is the result of the join. You can then load the resulting table into CAS.  
Perform the data preparation by interacting directly with your data source or by using the SQL procedure in SAS.
- 2 Establish a connection between the SAS client and your database by specifying a libref in a LIBNAME statement. You can then load your data into the SAS client using a DATA step.
- 3 Establish a connection between the SAS client and a CAS session by using a CAS LIBNAME statement. You can then load your data into a CAS table on the CAS server.
- 4 Perform analysis on your data in a CAS session. You can use the DATA step, procedures that work with CAS, and CAS actions on your CAS table. For more information, see [SAS Viya Quick Start](#) and [SAS Cloud Analytic Services: Fundamentals](#).
- 5 (Optional) You can load a CAS table that contains analysis results back into a SAS table using your CAS libref. If you want, you can load the SAS table back to your original database using your external database libref.

```
/* Add a libref to access a Greenplum database. */
libname mydblib greenplm server=mygpserver db=customers port=5432
      user=myuser password=gppwd;

/* Add a libref to the CAS server. */
libname c cas;

/* Load data into the SAS client. */
data mysasdat;
  set mydblib.myGPTab;
run;

/* Load SAS client data into a CAS session.          */
/* By default, the connection goes to the active caslib. */
data c.mycasdat;
  set mysasdat;
run;

/* Additional code to run analyses in CAS goes here. */

/* Optional: write CAS results back to the SAS client. */
data myreslts;
  set c.results;
run;
```

```
/* Optional: write the results out to the Greenplum database. */
data mydblib.results;
  set myreslts;
run;
```



# SAS Names and Support for DBMS Names

---

<b><i>DBMS-Specific Naming Conventions</i></b>	<b>22</b>
<b><i>SAS Naming Conventions</i></b>	<b>22</b>
Length of Name .....	22
Case Sensitivity .....	22
SAS Name Literals .....	23
<b><i>SAS/ACCESS Default Naming Behaviors</i></b>	<b>24</b>
Modification and Truncation of Column Names .....	24
ACCESS Procedure .....	24
DBLOAD Procedure .....	24
<b><i>Renaming DBMS Data</i></b>	<b>25</b>
Renaming SAS/ACCESS Tables .....	25
Renaming SAS/ACCESS Columns .....	25
<b><i>Options That Affect SAS/ACCESS Naming Behavior</i></b>	<b>25</b>
<b><i>Naming Behavior When Retrieving DBMS Data</i></b>	<b>26</b>
<b><i>Naming Behavior When Creating DBMS Objects</i></b>	<b>27</b>
<b><i>SAS/ACCESS Naming Examples</i></b>	<b>29</b>
Replacing Unsupported Characters .....	29
Preserving Column Names .....	29
Preserving Table Names .....	30
Using DQUOTE=ANSI .....	31
Using Name Literals .....	32
Using DBMS Data to Create a DBMS Table .....	33
Using a SAS Data Set to Create a DBMS Table .....	34

---

# DBMS-Specific Naming Conventions

Some DBMSs allow case-sensitive names and names with special characters. As a result, keep the considerations in this chapter in mind when you use the names of DBMS objects such as tables and columns with SAS/ACCESS features.

For information about how SAS handles your DBMS names, see the DBMS-specific reference section for your SAS/ACCESS interface.

---

# SAS Naming Conventions

---

## Length of Name

SAS naming conventions allow long names for SAS data sets and SAS variables. For example, MYDB TEMP\_EMPLOYEES\_QTR4\_2000 is a valid two-level SAS name for a data set.

Some SAS names can be up to 32 characters, depending on the SAS session encoding and the limits of the object name length of the DBMS. These SAS names can be up to 32 characters:

- members of SAS libraries, including SAS data sets, data views, catalogs, catalog entries, and indexes
- variables in a SAS data set
- macros and macro variables

These SAS language elements have a maximum length of eight characters:

- librefs and filerefs
- SAS engine names

For a description of SAS naming conventions, see “Rules for Most SAS Names” in *SAS Programmer’s Guide: Essentials*.

---

## Case Sensitivity

When SAS encounters mixed-case or case-sensitive names in SAS code, SAS stores and displays the names as they are specified. In the following example, two SAS variables (Flight and dates) are specified in mixed case.

```
input Flight $3. +3 dates date9.;
```

SAS then displays the variable names as specified, and the column headings appear as specified.

**Output 3.1 Mixed-Case Names Displayed in Output**

SAS System		
Obs	Flight	dates
1	114	01MAR2000
2	202	01MAR2000
3	204	01MAR2000

Although SAS stores variable names as they are specified, it recognizes variables for processing without regard to case. For example, SAS processes these variables as FLIGHT and DATES. Likewise, renaming the Flight variable to "flight" or "FLIGHT" would result in the same processing.

## SAS Name Literals

A SAS *name literal* is a name token that is expressed as a quoted string that is followed by the letter n. By using name literals, you can use special characters or blanks that are not otherwise allowed in SAS names when you specify a SAS data set or variable. Name literals are especially useful for expressing database column and tables names that contain special characters.

Here are two examples of name literals.

```
data mydblib.'My Staff Table'n;
data Budget_for_1999;
  input '$ Amount Budgeted'n 'Amount Spent'n;
```

Name literals are subject to certain restrictions.

- You can use a name literal only for SAS variable and data set names, statement labels, and DBMS column and table names.
- You can use a name literal only in a DATA step or in the SQL procedure.
- If a name literal contains any characters that are not allowed when VALIDVARNAME=V7, you must set the system option to VALIDVARNAME=ANY. For more information, see [VALIDVARNAME=](#) on page 686.

---

# SAS/ACCESS Default Naming Behaviors

---

## Modification and Truncation of Column Names

When SAS/ACCESS reads DBMS column names that contain characters that are not standard in SAS names, the default behavior is to replace an unsupported character with an underscore (\_). Nonstandard names include those with blank spaces or such special characters as @, #, % that are not allowed in SAS names. For example, the DBMS column name Amount Budgeted\$ becomes the SAS variable name Amount\_Budgeted\_.

When SAS/ACCESS encounters a DBMS column name that exceeds 32 characters, it truncates the name.

After it has modified or truncated a DBMS column name, SAS appends a number to the variable name, if necessary, to preserve uniqueness. For example, DBMS column names MY\$DEPT, My\$Dept, and my\$dept become SAS variable names MY\_DEPT, MY\_Dept0, and my\_dept1.

---

## ACCESS Procedure

If you try to use long names in the ACCESS procedure, you receive an error message that long names are not supported. Long member names, such as access descriptor and view descriptor names, are truncated to eight characters. Long DBMS column names are truncated to 8-character SAS variable names within the SAS access descriptor. You can use the [RENAME](#) statement to specify 8-character SAS variable names, or you can accept the default truncated SAS variable names that are assigned by the ACCESS procedure.

The ACCESS procedure converts DBMS object names to uppercase characters unless they are enclosed in quotation marks. Any DBMS objects that are given lowercase names when they are created, or whose names contain special or national characters, must be enclosed in quotation marks.

---

## DBLOAD Procedure

You can use long member names, such as the name of a SAS data set that you want to load into a DBMS table, in the DBLOAD procedure DATA= option. However, if you attempt to use long SAS variable names, you receive an error message advising you that long variable names are not supported in the DBLOAD procedure. You can use the [RENAME](#) to rename the 8-character SAS variable names to long DBMS column names when you load the data into a DBMS table. You can also use

the SAS data set option RENAME to rename the columns after they are loaded into the DBMS.

Most DBLOAD procedure statements convert lowercase characters in user-specified values and default values to uppercase. If your host or database is case sensitive and you want to specify a value that includes lowercase alphabetic characters (for example, a user ID or password), enclose the entire value in quotation marks. You must also put quotation marks around any value that contains special characters or national characters.

The only exception is the DBLOAD [SQL](#) statement. The DBLOAD SQL statement is passed to the DBMS exactly as you enter it with the case preserved.

---

## Renaming DBMS Data

---

### Renaming SAS/ACCESS Tables

You can rename DBMS tables and views using the CHANGE statement, as shown in this example.

```
proc datasets lib=x;
  change oldtable=newtable;
quit;
```

You can rename tables using this method for all SAS/ACCESS engines. However, if you change a table name, any view that depends on that table no longer works unless the view references the new table name.

---

### Renaming SAS/ACCESS Columns

You can use the [RENAME](#) statement to rename the 8-character default SAS variable names to long DBMS column names when you load the data into a DBMS table. You can also use the SAS data set option RENAME= to rename the columns after they are loaded into the DBMS.

---

## Options That Affect SAS/ACCESS Naming Behavior

To change how SAS handles case-sensitive or nonstandard DBMS table and column names, specify one or more of these options.

**PRESERVE\_COL\_NAMES=YES**

This option applies only to creating DBMS tables. When set to YES, it preserves spaces, special characters, and mixed case in DBMS column names. For more information, see the PRESERVE\_COL\_NAMES= [LIBNAME](#) and [data set](#) options.

*SAP HANA:* When this option is set to YES, you are allowed to use SAP HANA reserved words as column names.

**PRESERVE\_TAB\_NAMES=YES**

When set to YES, this option preserves blank spaces, special characters, and mixed case in DBMS table names. Specify the PRESERVE\_NAMES=YES | NO alias if you plan to specify both the PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options in your LIBNAME statement. Using this alias saves you time when you are coding. For more information, see the [PRESERVE\\_TAB\\_NAMES= LIBNAME option](#).

*SAP HANA:* When this option is set to YES, you are allowed to use SAP HANA reserved words as table names.

**DQUOTE=ANSI**

This PROC SQL option specifies whether PROC SQL treats values within double quotation marks as a character string, or as a column name or table name. When you specify DQUOTE=ANSI, your SAS code can refer to DBMS names that contain characters and spaces that SAS naming conventions do not allow. By specifying DQUOTE=ANSI, you can preserve special characters in table and column names in your SQL statements by enclosing the names in double quotation marks. To preserve table names, you must also specify PRESERVE\_TAB\_NAMES=YES. To preserve column names when you create a table, you must also specify PRESERVE\_COL\_NAMES=YES.

**VALIDMEMNAME=EXTEND**

This global system option allows you to include national characters and some special characters in a data set name. For more information, see [“VALIDMEMNAME= System Option” in SAS System Options: Reference](#).

**VALIDVARNAME=ANY**

This global system option can override SAS naming conventions. For more information, see the [VALIDVARNAME= system option on page 686](#).

**Examples** that use these options are available. The availability of these options and their default values are DBMS-specific, so see the SAS/ACCESS documentation for your DBMS to learn how the SAS/ACCESS engine for your DBMS processes names.

For general information about naming, see [“Rules for Most SAS Names” in SAS Programmer’s Guide: Essentials](#).

## Naming Behavior When Retrieving DBMS Data

The tables in this section illustrate how SAS/ACCESS processes DBMS names when it retrieves data from a DBMS. This information applies generally to all interfaces. However, in some cases you do not need to specify these options

because the option default values are DBMS-specific. For details, see the DBMS-specific reference section for your SAS/ACCESS interface. Available [examples](#) illustrate different types of naming actions and defaults.

**Table 3.1** DBMS Column Names to SAS Variable Names When Reading DBMS Data

DBMS Column Name	Desired SAS Variable Name	Options
Case-sensitive DBMS column name, such as Flight	Case-sensitive SAS variable name, such as Flight	No options are necessary.
DBMS column name with characters that are not valid in SAS names, such as My\$Flight	Case-sensitive SAS variable name where an underscore replaces the invalid characters, such as My_Flight	No options are necessary.
DBMS column name with characters that are not valid in SAS names, such as My\$Flight	Nonstandard, case-sensitive SAS variable name, such as My \$Flight	PROC SQL DQUOTE=ANSI or, in a DATA or PROC step, use a SAS name literal such as 'My \$Flight'n and VALIDVARNAME=ANY.

**Table 3.2** DBMS Table Names to SAS Data Set Names When Reading DBMS Data

DBMS Table Name	Desired SAS Data Set Name	Options
Default DBMS table name, such as STAFF	Default SAS data set or member name (uppercase), such as STAFF	PRESERVE_TAB_NAMES=NO
Case-sensitive DBMS table name, such as Staff	Case-sensitive SAS data set, such as Staff	PRESERVE_TAB_NAMES=YES
DBMS table name with characters that are not valid in SAS names, such as All\$Staff	Nonstandard, case-sensitive SAS data set name, such as All \$Staff	PROC SQL DQUOTE=ANSI and PRESERVE_TAB_NAMES=YES or, in a DATA step or PROC, use a SAS name literal such as 'All \$Staff'n and PRESERVE_TAB_NAMES=YES

## Naming Behavior When Creating DBMS Objects

The tables in this section illustrate how SAS/ACCESS handles variable names when it creates such DBMS objects as tables and views. This information applies

generally to all interfaces. However, in some cases you do not need to specify these options because the option default values are DBMS-specific. For details, see the documentation for your DBMS. Available [examples](#) illustrate different types of naming actions and defaults.

**Table 3.3** SAS Variable Names to DBMS Column Names When Creating Tables

SAS Variable Name as Input	Desired DBMS Column Name	Options
Any SAS variable name, such as Miles	Default DBMS column name (normalized to follow the DBMS's naming conventions), such as MILES	PRESERVE_COL_NAMES=NO
A case-sensitive SAS variable name, such as Miles	Case-sensitive DBMS column name, such as Miles	PRESERVE_COL_NAMES=YES
A SAS variable name with characters that are not valid in a normalized SAS name, such as Miles-to-Go	Case-sensitive DBMS column name that matches the SAS name, such as Miles-to-Go	PROC SQL DQUOTE=ANSI and PRESERVE_COL_NAMES=YES or, in a DATA or PROC step, use a SAS name literal and PRESERVE_COL_NAMES=YES and VALIDVARNAME=ANY

**Table 3.4** SAS Data Set Names to DBMS Table Names

SAS Data Set Name as Input	Desired DBMS Table Name	Options
Any SAS data set name, such as Payroll	Default DBMS table name (normalized to follow the DBMS's naming conventions), such as PAYROLL	PRESERVE_TAB_NAMES=NO
Case-sensitive SAS data set name, such as Payroll	Case-sensitive DBMS table name, such as Payroll	PRESERVE_TAB_NAMES=YES
Case-sensitive SAS data set name with characters that are not valid in a normalized SAS name, such as Payroll-for-QC	Case-sensitive DBMS table name that matches the SAS name, such as Payroll-for-QC	PROC SQL DQUOTE=ANSI and PRESERVE_TAB_NAMES=YES or, in a DATA or PROC step, use a SAS name literal and PRESERVE_TAB_NAMES=YES

---

# SAS/ACCESS Naming Examples

---

## Replacing Unsupported Characters

This example creates the view, Myview, from the Oracle table, Mytable.

```
proc sql;
  connect to oracle (user=myusr1 password=mypwd1);
  create view myview as
    select * from connection to oracle
      (select "Amount Budgeted$", "Amount Spent$"
       from mytable);
  quit;

  proc contents data=myview;
  run;
```

In the output that PROC CONTENTS produces, Oracle column names that the SQL View of MYTABLE processed are renamed to different SAS variable names: Amount Budgeted\$ becomes Amount\_Budgeted\_ and Amount Spent\$ becomes Amount\_Spent\_.

---

## Preserving Column Names

In this example, the Oracle table, PAYROLL, creates a new Oracle table, PAY1, which it then prints. You can use both PRESERVE\_COL\_NAMES=YES and the PROC SQL DQUOTE=ANSI options to preserve the case and nonstandard characters in the column names. You do not need to quote the column aliases to preserve mixed case. You need only double quotation marks when the column name has nonstandard characters or blanks.

By default, most SAS/ACCESS interfaces use DBMS-specific rules to specify the case of table and column names. So, even though the new pay1 Oracle table name in this example is created in lowercase, Oracle stores the name in uppercase as PAY1. To store the table name as "pay1", specify PRESERVE\_TAB\_NAMES=NO.

```
options linesize=120 pagesize=60 nodate;

libname mydblib oracle user=myusr1 password=mypwd1 path='mysrv1'
  schema=hrdept preserve_col_names=yes;

proc sql dquote=ansi;
  create table mydblib.pay1 as
    select idnum as "ID #", sex, jobcode, salary,
           birth as BirthDate, hired as HiredDate
      from mydblib.payroll
     order by birth;
```

```
title "Payroll Table with Revised Column Names";
select * from mydblib.pay1;
quit;
```

SAS recognizes the JOBCODE, SEX, and SALARY column names, regardless of how you specify them in your SAS code: as lowercase, mixed case, or uppercase. SEX, JOBCODE, and SALARY columns in the PAYROLL Oracle table were created in uppercase. So they retain this case in the new table unless you rename them. Here is partial output from the example.

**Output 3.2** DBMS Table Created with Nonstandard and Standard Column Names

Payroll Table with Revised Column Names						
ID #	SEX	JOBCODE	SALARY	BirthDate	HiredDate	
1118	M	PT3	11379	16JAN1944:00:00:00	18DEC1980:00:00:00	
1065	M	ME2	35090	26JAN1944:00:00:00	07JAN1987:00:00:00	
1409	M	ME3	41551	19APR1950:00:00:00	22OCT1981:00:00:00	
1401	M	TA3	38822	13DEC1950:00:00:00	17NOV1985:00:00:00	
1890	M	PT2	91908	20JUL1951:00:00:00	25NOV1979:00:00:00	

## Preserving Table Names

This example uses PROC PRINT to print the DBMS table PAYROLL. The DBMS table was created in uppercase. Because PRESERVE\_TAB\_NAMES=YES, you must specify the table name in uppercase. (If you specify PRESERVE\_TAB\_NAMES=NO, you can specify the DBMS table name in lowercase.) Partial output follows the example.

```
options nodate linesize=64;
libname mydblib oracle user=myusr1 password=mypwd1
path='mysrv1' preserve_tab_names=yes;

proc print data=mydblib.PAYROLL;
title 'PAYROLL Table';
run;
```

**Output 3.3** DBMS Table with a Case-Sensitive Name

PAYROLL Table					
Obs	IDNUM	SEX	JOBCODE	SALARY	BIRTH
1	1919	M	TA2	34376	12SEP1960:00:00:00
2	1653	F	ME2	35108	15OCT1964:00:00:00
3	1400	M	ME1	29769	05NOV1967:00:00:00
4	1350	F	FA3	32886	31AUG1965:00:00:00
5	1401	M	TA3	38822	13DEC1950:00:00:00

## Using DQUOTE=ANSI

This example creates a DBMS table with a blank space in its name. It uses double quotation marks to specify the table name, International Delays. You can also specify both of the preserve names LIBNAME options by using the alias PRESERVE\_NAMES=. Because PRESERVE NAMES=YES, the schema airport is now case sensitive for Oracle.

```
options linesize=64 nodate;

libname mydblib oracle user=myusr1 password=mypwd1 path='airdata'
      schema=airport preserve_names=yes;

proc sql dquote=ansi;
create table mydblib."International Delays" as
  select int.flight as "FLIGHT NUMBER", int.dates,
         del.orig as ORIGIN,
         int.dest as DESTINATION, del.delay
    from mydblib.INTERNAT as int,
         mydblib.DELAY as del
   where int.dest=del.dest and int.dest='LON';
quit;

proc sql dquote=ansi outobs=10;
  title "International Delays";
  select * from mydblib."International Delays";
```

You can use single quotation marks to specify the data value for London (int.dest='LON') in the WHERE clause. Because of the preserve name LIBNAME options, using double quotation marks would cause SAS to interpret this data value as a column name.

**Output 3.4** DBMS Table with Nonstandard Column Names

International Delays				
FLIGHT NUMBER	DATES	ORIGIN	DESTINATION	DELAY
219	01MAR1998:00:00:00	LGA	LON	18
219	02MAR1998:00:00:00	LGA	LON	18
219	03MAR1998:00:00:00	LGA	LON	18
219	04MAR1998:00:00:00	LGA	LON	18
219	05MAR1998:00:00:00	LGA	LON	18
219	06MAR1998:00:00:00	LGA	LON	18
219	07MAR1998:00:00:00	LGA	LON	18
219	01MAR1998:00:00:00	LGA	LON	18
219	02MAR1998:00:00:00	LGA	LON	18
219	03MAR1998:00:00:00	LGA	LON	18

Using a label to change the name of a DBMS column name changes only the output. Enclose the label in single quotation marks. Because this column name and the table name (International Delays) each contain a space in their names, you must enclose the names in double quotation marks. Partial output follows the example.

```

options linesize=64 nodate;

libname mydblib oracle user=myusr1 password=mypwd1 path='airdata'
    schema=airport preserve_names=yes;

proc sql dquote=ansi outobs=5;
    title "Query from International Delays";
    select "FLIGHT NUMBER" label='Flight_Number', dates, delay
        from mydblib."International Delays";

```

**Output 3.5** Query Renaming a Nonstandard Column to a Standard SAS Name

Query from International Delays		
Flight_Number	DATES	DELAY
219	01MAR1998:00:00:00	18
219	02MAR1998:00:00:00	18
219	03MAR1998:00:00:00	18
219	04MAR1998:00:00:00	18
219	05MAR1998:00:00:00	18

You can preserve special characters by specifying DQUOTE=ANSI and using double quotation marks around the SAS names in your SELECT statement.

```

proc sql dquote=ansi;
    connect to oracle (user=myusr1 password=mypwd1);
    create view myview as
        select "Amount Budgeted$", "Amount Spent$"
        from connection to oracle
            (select "Amount Budgeted$", "Amount Spent$"
            from mytable);
    quit;
    proc contents data=myview;
    run;

```

Output from this example would show that Amount Budgeted\$ remains Amount Budgeted\$ and Amount Spent\$ remains Amount Spent\$.

## Using Name Literals

This example creates a table using name literals. To use name literals, you must specify the SAS option VALIDVARNAME=ANY. Use PROC SQL to print the new DBMS table because name literals work only with PROC SQL and the DATA step. PRESERVE\_COL\_NAMES=YES is required only because the table is being created with nonstandard SAS column names.

```

options ls=64 validvarname=any nodate;

libname mydblib oracle user=myusr1 password=mypwd1 path='mysrv1'
    preserve_col_names=yes preserve_tab_names=yes ;

data mydblib.'Sample Table'n;
    'EmpID#'n=12345;

```

```

Lname='Chen';
'Salary in $'n=63000;

proc sql;
  title "Sample Table";
  select * from mydblib.'Sample Table'n;

```

**Output 3.6 DBMS Table to Test Column Names**

Sample Table		
EmpID#	Lname	Salary in \$
12345	Chen	63000

## Using DBMS Data to Create a DBMS Table

This example uses PROC SQL to create a DBMS table that is based on data from other DBMS tables. To preserve the case sensitivity of the aliased column names, use PRESERVE\_COL\_NAMES=YES. Partial output follows the example.

```

libname mydblib oracle user=myusr1 password=mypwd1
  path='hrdata99' schema=personnel preserve_col_names=yes;

proc sql;
create table mydblib.gtforty as
  select lname as LAST_NAME,
         fname as FIRST_NAME,
         salary as ANNUAL_SALARY
    from mydblib.staff a,
         mydblib.payroll b
   where (a.idnum eq b.idnum) and
         (salary gt 40000)
  order by lname;

proc print noobs;
  title 'Employees with Salaries Greater Than $40,000';
run;

```

**Output 3.7 Updating DBMS Data**

Employees with Salaries Greater Than \$40,000		
LAST_NAME	FIRST_NAME	ANNUAL_SALARY
BANADYGA	JUSTIN	88606
BAREFOOT	JOSEPH	43025
BRADY	CHRISTINE	68767
BRANCACCIO	JOSEPH	66517
CARTER-COHEN	KAREN	40260
CASTON	FRANKLIN	41690
COHEN	LEE	91376
FERNANDEZ	KATRINA	51081

## Using a SAS Data Set to Create a DBMS Table

This example uses a SAS DATA step to create a DBMS table, College-Hires-1999, from a temporary SAS data set that has case-sensitive names. It creates the temporary data set and specifies the LIBNAME statement. Because it uses a DATA step to create the DBMS table, it must specify the table name as a name literal and specify the PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options. In this case, it performs these actions by using the alias PRESERVE\_NAMES=.

```

options validvarname=any nodate;

data College_Hires_1999;
    input IDnum $4. +3 Lastname $11. +2
          Firstname $10. +2 City $15. +2
          State $2. ;
    datalines;
3413    Schwartz      Robert        New Canaan     CT
3523    Janssen       Heike        Stamford      CT
3565    Gomez         Luis         Darien       CT
;

libname mydblib oracle user=myusr1 password=mypwd1
    path='hrdata99' schema=hrdept preserve_names=yes;

data mydblib.'College-Hires-1999'n;
    set College_Hires_1999;

proc print;
    title 'College Hires in 1999';
run;

```

**Output 3.8 DBMS Table with Case-Sensitive Table and Column Names**

College Hires in 1999					
Obs	IDnum	Lastname	Firstname	City	State
1	3413	Schwartz	Robert	New Canaan	CT
2	3523	Janssen	Heike	Stamford	CT
3	3565	Gomez	Luis	Darien	CT



# Data Integrity and Security

---

<b>DBMS Security .....</b>	<b>37</b>
Privileges .....	37
Triggers .....	38
<b>SAS Security .....</b>	<b>38</b>
Securing Data .....	38
Assigning SAS Passwords .....	38
Protecting Connection Information .....	40
Extracting DBMS Data to a SAS Data Set .....	40
Specifying Views and Schemas .....	41
Controlling DBMS Connections .....	41
Locking, Transactions, and Currency Control .....	42
Customizing a DBMS Connection .....	43
<b>Potential Result Set Differences When Processing Null Data .....</b>	<b>43</b>

---

## DBMS Security

---

### Privileges

Database administrators control who has privileges to access or update DBMS objects. They also control who can create objects, and object creators control who can access the objects. Users cannot use DBMS facilities to access DBMS objects through SAS/ACCESS software unless they have the appropriate DBMS privileges or authority on those objects. You can grant privileges on the DBMS side by using the [SQL pass-through facility](#) to EXECUTE an SQL statement or by issuing a GRANT statement from the [DBLOAD procedure](#).

You should give users only the privileges on the DBMS that they must have. Privileges are granted on whole tables or views. You must explicitly grant user privileges on the DBMS tables or views that underlie a view so that users can use that view.

For more information about ensuring security on the DBMS side of the interface, see your DBMS documentation.

---

## Triggers

If your DBMS supports triggers, you can use them to enforce security authorizations or business-specific security considerations. Triggers are executed based on when the SQL statement is executed and how often the trigger is executed. Triggers can be executed before an SQL statement is executed, after an SQL statement is executed, or for each row of an SQL statement. Also, triggers can be specified for DELETE, INSERT, and UPDATE statement execution.

Enabling triggers can provide more specific security for Delete, Insert, and Update operations. SAS/ACCESS abides by all constraints and actions that are specified by a trigger. For more information, see the documentation for your DBMS.

---

## SAS Security

---

### Securing Data

SAS preserves the data security provided by your DBMS and operating system. However, SAS/ACCESS does not override the security of your DBMS. To secure DBMS data from accidental update or deletion, from the SAS side of the interface, you can take steps like these.

- Specify the SAS/ACCESS **DBPROMPT= LIBNAME** option to avoid saving connection information in your code.
- Create SQL views and protecting them from unauthorized access by applying passwords.

These and other approaches are discussed in detail in subsequent sections.

---

### Assigning SAS Passwords

By using SAS passwords, you can protect SQL views, SAS data sets, and descriptor files from unauthorized access. The following table summarizes the levels of protection that SAS passwords provide. Note that you can assign multiple levels of protection.

**Table 4.1** Password Protection Levels and Their Effects

File Type	READ=	WRITE=	ALTER=
PROC SQL view of DBMS data	Protects the underlying data from being read or updated through the view; does not protect against replacement of the view	Protects the underlying data from being updated through the view; does not protect against replacement of the view	Protects the view from being modified, deleted, or replaced

You can use these methods to assign, change, or delete a SAS password:

- the global SETPASSWORD command, which opens a dialog box
- the DATASETS procedure's MODIFY statement

Here is the syntax for using PROC DATASETS to assign a password to a SAS data set.

```
PROC DATASETS LIBRARY=libref MEMTYPE=member-type;
MODIFY member-name (password-level = password-modification);
RUN;
```

The *password-level* argument can have one or more of these values: READ=, WRITE=, ALTER=, or PW=. PW= assigns Read, Write, and Alter privileges to a data file. The *password-modification* argument enables you to assign a new password or to change or delete an existing password. For example, this PROC DATASETS statement assigns the password MONEY with the ALTER level of protection to the data in Adlib.Salaries:

```
proc datasets library=adlib;
  modify salaries (alter=money);
  run;
```

In this case, users are prompted for the password whenever they try to browse or update the data or try to create new data sets that are based on Adlib.Salaries.

In the next example, the PROC DATASETS statement assigns the passwords MYPW and MYDEPT with READ and ALTER levels of protection to the data in Mylib.Jobc204:

```
proc datasets library=mylib;
  modify jobc204 (read=mypw alter=mydept);
  run;
```

In this case, users are prompted for the SAS password when they try to read the DBMS data or try to browse or update the data in Vlib.Jobc204. A user could still update the data in Vlib.Jobc204. For example, a user could update the data by using a PROC SQL UPDATE statement. (To prevent user updates, assign a WRITE level of protection to prevent data updates.)

When you assign multiple levels of passwords, use a different password for each level to ensure that you grant only the access privileges that you intend.

To delete a password, put a slash after the password:

```
proc datasets library=mylib;
  modify jobc204 (read=mypw/ alter=mydept/);
```

```
run;
```

---

## Protecting Connection Information

In addition to directly controlling access to data, you can protect the data indirectly by protecting the connection information that SAS/ACCESS uses to reach the DBMS. Generally, you can achieve this by not saving connection information in your code.

One way to protect connection information is by storing user name, password, and other connection options in a local environment variable. Access to the DBMS is denied unless the correct user and password information is stored in a local environment variable. See the documentation for your DBMS to determine whether this alternative is supported.

For Hadoop, another way to protect connection information between Hive and SAS/ACCESS Interface to Hadoop is to use Kerberos authentication. When you use Kerberos, there are no sensitive user names and passwords to protect. For more information, see “[Hadoop Configuration](#)” on page 897.

Another way to protect connection information is by requiring users to manually enter it at connection time. When you specify **DBPROMPT=YES** in a SAS/ACCESS LIBNAME statement, each user has to provide DBMS connection information in a dynamic, interactive manner. The statement below opens a dialog box to prompt the user to enter connection information, such as a user name and password:

```
libname myoralib oracle dbprompt=yes defer=no;
```

The dialog box that appears contains the DBMS connection options that are valid for the SAS/ACCESS engine that is being used. In the preceding LIBNAME statement, the connection options are for Oracle.

Using the **DBPROMPT=** option in the LIBNAME statement offers several advantages. DBMS account passwords are protected because they do not need to be stored in a SAS program or descriptor file. Also, when a password or user name changes, the SAS program does not need to be modified. Another advantage is that the same SAS program can be used by any valid user name and password combination that is specified during execution. You can also use connection options in this interactive manner when you want to run a program on a production server instead of testing a server without modifying your code. By using the prompt window, you can specify the new server name dynamically.

See the section “[DBPROMPT= Data Set Option](#)” on page 526 for more information about which engines support the **DBPROMPT= LIBNAME** option.

---

## Extracting DBMS Data to a SAS Data Set

If you extract data from a SAS view with an assigned SAS password, the new SAS data set is automatically assigned to the same password. If a view does not have a password, you can assign a password to the extracted SAS data set by using the MODIFY statement in the DATASETS procedure. For more information, see the [Base SAS Procedures Guide](#).

---

## Specifying Views and Schemas

If you want to provide access to some but not all fields in a DBMS table, create a SAS view that prohibits access to the sensitive data. To restrict access to some columns, specify that those columns be dropped. Columns that are dropped from views do not affect the underlying DBMS table and can be reselected for later use.

Some SAS/ACCESS engines support LIBNAME options that restrict or qualify the scope, or schema, of the tables in the libref. For example, the DB2 engine supports the [AUTHID=](#) and [LOCATION=](#) options. The Oracle engine supports the [SCHEMA=](#) and [DBLINK=](#) options. See the SAS/ACCESS documentation for your DBMS to determine which options are available to you.

This example uses SAS/ACCESS Interface to Oracle.

```
libname myoralib oracle user=myusr1 password=mypwd1
      path='mysrv1' schema=testgroup;

proc datasets lib=mysrv1;
run;
```

In this example, the MYORALIB libref is associated with the Oracle schema named TESTGROUP. The DATASETS procedure lists only the tables and views that are accessible to the TESTGROUP schema. Any reference to a table that uses the MYORALIB libref is passed to the Oracle server as a qualified table name. For example, if the SAS program reads a table by specifying the SAS data set MYSRV1.TESTTABLE, the SAS/ACCESS engine passes this query to the server.

```
select * from "testgroup.testtable"
```

---

## Controlling DBMS Connections

SAS/ACCESS supports the [CONNECTION=](#) and [DEFER=](#) options to control when a DBMS connection is made, and how many connections are executed within the context of your SAS/ACCESS application. For most SAS/ACCESS engines, a connection to a DBMS begins one transaction, or work unit, and all statements issued in the connection execute within the context of the active transaction.

The CONNECTION= LIBNAME option enables you to specify how many connections are executed when the library is used and which operations on tables are shared within a connection. For many DBMSs, the default value is CONNECTION=SHAREDREAD, which means that a SAS/ACCESS engine executes a *shared read* DBMS connection when the library is assigned. Every time a table in the library is read, the read-only connection is used. However, if an application attempts to update data using the libref, a separate connection is issued, and the update occurs in the new connection. As a result, there is one connection for read-only transactions and a separate connection for each update transaction.

In the example below, the SAS/ACCESS engine issues a connection to the DBMS when the libref is assigned. The PRINT procedure reads the table by using the first connection. When PROC SQL queries the table, the query uses a second connection to the DBMS.

```

libname myhadlib hadoop server=hadoopsrv user=myusr1 password=mypwd1;

proc print data=myhadlib.mytable;
run;

proc sql;
  select date1 from myhadlib.mytable;
quit;

```

This example uses SAS/ACCESS Interface to DB2 under z/OS. The LIBNAME statement executes a connection by way of the DB2 Call Attach Facility to the DB2 DBMS server.

```
libname mydb2lib db2 authid=myusr1;
```

To assign more than one SAS libref to your DBMS server when you do not plan to update the DBMS tables, SAS/ACCESS lets you optimize how the engine makes connections. Your SAS librefs can share a single read-only connection to the DBMS if you use the CONNECTION=GLOBALREAD option. This example shows how to use the CONNECTION= option with the ACCESS= option to control your connection and to specify read-only data access.

```
libname mydblib1 db2 authid=myusr1
  connection=globalread access=readonly;
```

If you do not want a connection to occur when a library is assigned, you can delay the connection to the DBMS by using the DEFER= option. When you specify DEFER=YES in the LIBNAME statement, the SAS/ACCESS engine connects to the DBMS the first time a DBMS object is referenced in a SAS program:

```
libname myhadlib hadoop server=hadoopsrv user=user1 pass=mypwd1 defer=yes;
```

---

**Note:** If you use DEFER=YES to assign librefs to your DBMS tables and views in an AUTOEXEC program, the processing of the AUTOEXEC file is faster. The processing is faster because the connections to the DBMS are not made every time SAS is invoked.

---

## Locking, Transactions, and Currency Control

SAS/ACCESS provides options so that you can control some of the row, page, or table locking operations that the DBMS and SAS/ACCESS engine perform as your programs are executed. For example, by default, the SAS/ACCESS Oracle engine does not lock any data when it reads rows from Oracle tables. However, you can override this behavior by using the [locking options](#) that SAS/ACCESS Interface to Oracle supports.

To lock the data pages of a table while SAS is reading the data to prevent other processes from updating the table, use the [READLOCK\\_TYPE=](#) option, as shown in this example.

```

libname myoralib oracle user=myusr1 pass=mypwd1
  path='mysrv1' readlock_type=table;

data work.mydata;
  set myoralib.mytable(where=(colnum > 123));

```

```
run;
```

Here the SAS/ACCESS Oracle engine obtains a TABLE SHARE lock on the table so that other processes cannot update the data while your SAS program reads it.

In this next example, Oracle acquires row-level locks on rows read for update in the tables in the libref.

```
libname myorplib oracle user=myusr1 password=mypwd1
path='mysrv1' updatelock_type=row;
```

Each SAS/ACCESS interface supports specific options. See the DBMS-specific reference section for your SAS/ACCESS interface to determine which options it supports.

## Customizing a DBMS Connection

To specify DBMS commands or stored procedures to run immediately after a DBMS connection, use the **DBCONINIT= LIBNAME** option. Here is an example.

```
libname myorplib oracle user=myusr1 password=mypwd1
path='mysrv1' dbconinit="EXEC MY_PROCEDURE";

proc sql;
update myorplib.mytable set acctnum=123
where acctnum=567;
quit;
```

When a libref is assigned, the SAS/ACCESS engine connects to the DBMS and passes a command that you specify to the DBMS to execute the stored procedure **MY\_PROCEDURE**. By default, a new connection to the DBMS is made for every table that is opened for updating. Therefore, the command is executed an additional time after a connection is made to update the table **MYTABLE**.

To execute a DBMS command or stored procedure only after the first connection in a library assignment, you can use the **DBLIBINIT=** option. Similarly, you can use the **DBLIBTERM= LIBNAME** option to specify a command to run before the disconnection of only the first library connection. Here is an example.

```
libname myorplib oracle user=myusr1 password=mypwd1
dblibinit="EXEC MY_INIT" dblibterm="EXEC MY_TERM";
```

## Potential Result Set Differences When Processing Null Data

When your data contains null values or when internal processing generates intermediate data sets that contain null values, you might receive different results depending on whether SAS or the DBMS does the processing. Although in many cases this does not present a problem, it is important to understand how these differences occur.

Most relational database systems have a special value called null, which means an absence of information and is analogous to a SAS missing value. SAS/ACCESS translates SAS missing values to DBMS null values when creating DBMS tables from within SAS. Conversely, SAS/ACCESS translates DBMS null values to SAS missing values when reading DBMS data into SAS.

However, there is an important difference in the behavior of DBMS null values and SAS missing values.

- A DBMS null value is interpreted as the absence of data, so you cannot sort a DBMS null value or evaluate it with standard comparison operators.
- A SAS missing value is interpreted as its internal floating-point representation because SAS supports 28 missing values (where a period (.) is the most common missing value). Because SAS supports multiple missing values, you can sort a SAS missing value and evaluate it with standard comparison operators.

This means that SAS and the DBMS interpret null values differently, which has significant implications when SAS/ACCESS passes queries to a DBMS for processing. This can be an issue in these situations:

- when filtering data (for example, in a WHERE clause, a HAVING clause, or an outer join ON clause). SAS interprets null values as missing; many DBMS exclude null values from consideration. For example, if you have null values in a DBMS column that is used in a WHERE clause, your results might differ depending on whether the WHERE clause is processed in SAS or is passed to the DBMS for processing. This is because the DBMS removes null values from consideration in a WHERE clause, but SAS does not.
- when using certain functions. For example, if you use the MIN aggregate function on a DBMS column that contains null values, the DBMS does not consider the null values, but SAS interprets the null values as missing. This interpretation affects the result.
- when submitting outer joins where internal processing generates nulls for intermediate result sets.
- when sorting data. SAS sorts null values low; most DBMSs sort null values high. For more information, see “[Sorting DBMS Data](#)” on page 49.

For example, create a simple data set that consists of one observation and one variable.

```
libname myhadlib hadoop user=myusr1 password=mypwd1;
data myhadlib.table;
x=.;      /*create a missing value */
run;
```

Then print the data set using a WHERE clause, which SAS/ACCESS passes to the DBMS for processing.

```
proc print data=myhadlib.table;
  where x<0;
run;
```

The log indicates that the WHERE clause selected no observations. Hadoop interprets the missing value as the absence of data and does not evaluate it with the less-than (<) comparison operator.

When there is the potential for inconsistency, consider using one of these strategies.

- Use the [DIRECT\\_SQL= LIBNAME](#) option to control whether SAS or the DBMS handles processing.

- Use the [SQL pass-through facility](#) to ensure that the DBMS handles processing.
- Add the *is not null* expression to WHERE clauses and ON clauses to ensure that you obtain the same result regardless of whether SAS or the DBMS does the processing.

Use the **NULLCHAR=** data set option to specify how the DBMS interprets missing SAS character values when updating DBMS data or inserting rows into a DBMS table.

You can use the first of these strategies to force SAS to process the data in this example.

```
libname myhadlib hadoop user=myusr1 password=mypwd1  
      direct_sql=nowhere; /* forces SAS to process WHERE clauses */  
  
data myhadlib.table;  
  x=.; /*create a missing value */  
run;
```

You can then print the data set using a WHERE clause

```
proc print data=myhadlib.table;  
  where x<0;  
run;
```

This time the log indicates that one observation was read from the data set because SAS evaluates the missing value as satisfying the less-than-zero condition in the WHERE clause.



# Performance Considerations

---

<i>Increasing Throughput of the SAS Server</i> .....	47
<i>Limiting Retrieval</i> .....	48
Row and Column Selection .....	48
The KEEP= and DROP= Options .....	48
<i>Repeatedly Accessing Data</i> .....	49
<i>Sorting DBMS Data</i> .....	49
<i>Temporary Table Support for SAS/ACCESS</i> .....	51
Overview .....	51
General Temporary Table Use .....	52
Pushing Heterogeneous Joins .....	52
Pushing Updates .....	53

---

## Increasing Throughput of the SAS Server

When you start SAS as a server that responds to multiple clients, you can use the [DBSRVTP= system option](#) to improve the performance of the clients. This option tells the SAS server whether to put a hold (block) on the originating client while making performance-critical calls to the database. By holding or blocking the originating client, the SAS/ACCESS server remains available for other clients; they do not need to wait for the originating client to complete its call to the database.

# Limiting Retrieval

## Row and Column Selection

Limiting the number of rows that the DBMS returns to SAS is an extremely important performance consideration. The less data that the SAS job requests, the faster the job runs.

Wherever possible, specify selection criteria that limit the number of rows that the DBMS returns to SAS. Use the SAS WHERE clause to retrieve a subset of the DBMS data.

If you are interested in only the first few rows of a table, consider adding the OBS= option. SAS passes this option to the DBMS to limit the number of rows to transmit across the network. Using the OBS= option can significantly improve performance for larger tables. To do this if you are using SAS Enterprise Guide, select **View** ⇒ **Explorer**, select the table that you want from the list of tables, and select the member that you want to see the contents of the table.

Likewise, select only the DBMS columns that your program needs. Selecting unnecessary columns slows your job.

## The KEEP= and DROP= Options

Just as with a SAS data set you can use the DROP= and KEEP= data set options to prevent retrieving unneeded columns from your DBMS table.

In this example, the KEEP= data set option causes the SAS/ACCESS engine to select only the SALARY and DEPT columns when it reads the Employees table.

```
libname myhadlib hadoop user=user1 password=mypwd1 server=hadoopsrv;

proc print data=myhadlib.employees (keep=salary dept);
  where dept='ACC024';
quit;
```

The DBMS generates SQL that is similar to this:

```
SELECT "SALARY", "DEPT" FROM EMPLOYEES
  WHERE (DEPT="ACC024")
```

Without the KEEP option, the DBMS processes SQL that is similar to this code:

```
SELECT * FROM EMPLOYEES WHERE (DEPT="ACC024")
```

This results in all columns from the Employees table being read in to SAS.

The DROP= data set option is a parallel option that specifies columns to omit from the output table. Keep in mind that the DROP= and KEEP= data set options are not interchangeable with the DROP and KEEP statements. Use of the DROP and KEEP

statements when selecting data from a DBMS can result in retrieval of all columns into SAS. This can seriously impact performance.

For example, this code results in all columns from the Employees table being retrieved into SAS. When creating the output data set, the KEEP statement is applied.

```
libname myhadlib hadoop user=testid password=testpass server=hadoopsrv;

data temp;
  set myhadlib.employees;
  keep salary;
run;
```

Here is how you can use the KEEP= data set option to retrieve only the SALARY column.

```
data temp;
  set myhadlib.employees(keep=salary);
run;
```

---

## Repeatedly Accessing Data

---

### **CAUTION**

If you need to access the most current DBMS data, access it directly from the database every time. Do not follow the extraction suggestions in this section.

---

It is sometimes more efficient to extract (copy) DBMS data to a SAS data set than to repeatedly read the data. SAS data sets are organized to provide optimal performance with PROC and DATA steps. Programs that use SAS data sets often outperform SAS programs that read DBMS data.

Consider extracting data when you work with a large DBMS table and plan to use the same DBMS data in several procedures or DATA steps.

You can extract DBMS data to a SAS data set by using the DATA step or the OUT= option in a SAS procedure.

---

## Sorting DBMS Data

---

Sorting DBMS data can be resource-intensive—whether you use the SORT procedure, a BY statement, or an ORDER BY clause on a DBMS data source or in the SQL procedure SELECT statement. Sort data only when it is needed for your program.

Here are guidelines for sorting data.

- If you specify a BY statement in a DATA or PROC step that references a DBMS data source, it is recommended for performance reasons that you associate the BY variable with an indexed DBMS column. If you reference DBMS data in a

SAS program and the program includes a BY statement for a variable that corresponds to a column in the DBMS table, the SAS/ACCESS LIBNAME engine automatically generates an ORDER BY clause for that variable. The ORDER BY clause causes the DBMS to sort the data before the DATA or PROC step uses the data in a SAS program. If the DBMS table is very large, this sorting can adversely affect your performance. Use a BY variable that is based on an indexed DBMS column in order to reduce this negative impact.

- The outermost BY or ORDER BY clause overrides any embedded BY or ORDER BY clauses. This includes those specified by the **DBCONDITION=** option, in a WHERE clause, and in the selection criteria in a view descriptor. In the following example, the Exec\_Employees data set includes a BY statement that sorts data by the HIREDATE variable. However, when that data set is used in the following PROC SQL query, the data is ordered by the SALARY column, not by HIREDATE.

```
libname mydblib hadoop server=hadoopsvr database=compdata user=user1
      pass=mypwd1;
data exec_employees;
  set mydblib.employees (keep=lastname firstname empid salary hiredate);
  by hiredate;
  where salary >= 75000;
run;

proc sql;
  select * from exec_employees
    order by salary;
quit;
```

- Do not use PROC SORT to sort data from SAS back into the DBMS because this impedes performance and has no effect on the order of the data.
- Do not use the SORTSEQ= system option because this option has no effect on the sort order of the data. Whenever possible, SAS allows sorting to be performed by the DBMS to improve performance. The values for the SORTSEQ= system option apply to processing that is performed by SAS. Therefore, the option has no impact when data is sorted by the DBMS.
- The database does not guarantee sort stability when you use PROC SORT. Sort stability means that the ordering of the observations in the BY statement is exactly the same every time the sort is run against static data. If you absolutely require sort stability, you must place your database data into a SAS data set and use PROC SORT.
- When you use PROC SORT, be aware that the sort rules for SAS and for your DBMS might be different. Use the SAS system option SORTPGM to specify which rules (host, SAS, or DBMS) are applied:

**SORTPGM=BEST**

sorts data according to the DBMS sort rules, the host sort rules, and the SAS sort rules. (Sorting uses the first available and pertinent sorting algorithm in this list.) This is the default.

**SORTPGM=HOST**

sorts data according to host rules and then SAS rules. (Sorting uses the first available and pertinent sorting algorithm in this list.)

**SORTPGM=SAS**

sorts data by SAS rules.

---

# Temporary Table Support for SAS/ACCESS

---

## Overview

DBMS temporary table support in SAS consists of the ability to retain DBMS temporary tables from one SAS step to the next. This ability is a result of establishing a SAS connection to the DBMS that persists across multiple SAS procedures and DATA steps.

Temporary table support is available for these DBMSs.

*Table 5.1 DBMS-Specific Temporary Table Support*

DBMS	Temporary Table Support	DBMSTEMP= LIBNAME Support
Aster	yes	yes
DB2	yes	yes
Greenplum	yes	yes
Hadoop	yes	yes
Impala	no	no
Informix	yes	no
JDBC	yes	yes
Microsoft SQL Server	yes	yes
MySQL	yes	yes
Netezza	yes	yes
ODBC	yes	yes
OLE DB	yes	yes
Oracle	yes	yes

DBMS	Temporary Table Support	DBMSTEMP= LIBNAME Support
PostgreSQL	yes	yes
SAP ASE	yes	no <sup>1</sup>
SAP HANA	yes	yes
SAP IQ	yes	yes
Snowflake	yes	yes
Spark	no	no
Teradata	yes	yes
Vertica	yes	yes
Yellowbrick	yes	yes

<sup>1</sup> The SAP ASE engine uses the same special table-name syntax as the native database to create a temporary table.

The value of DBMS temporary table support in SAS is increased performance potential. By pushing the processing to the DBMS in certain situations, you can achieve an overall performance gain. The processes in this section provide a general outline of how to use DBMS temporary tables.

## General Temporary Table Use

Follow these steps to use temporary tables on the DBMS.

- 1 Establish a global connection to the DBMS that persists across SAS procedure and DATA step boundaries.
- 2 Create a DBMS temporary table and load it with data.
- 3 Use the DBMS temporary table with SAS.

Closing the global connection causes the DBMS temporary table to close as well.

## Pushing Heterogeneous Joins

Follow these steps to push heterogeneous joins to the DBMS.

- 1 Establish a global connection to the DBMS that persists across SAS procedure and DATA step boundaries.
- 2 Create a DBMS temporary table and load it with data.

- 3 Perform a join on the DBMS using the DBMS temporary and DBMS permanent tables.
- 4 Process the result of the join with SAS.

---

## Pushing Updates

To push updates (process transactions) to the DBMS:

- 1 Establish a global connection to the DBMS that persists across SAS procedure and DATA step boundaries.
- 2 Create a DBMS temporary table and load it with data.
- 3 Issue SQL that uses values in the temporary table to process against the production table.
- 4 Process the updated DBMS tables with SAS.

Although these processing scenarios are purposely generic, they apply to each DBMS that supports temporary tables. For details, see the “[DBMSTEMP= LIBNAME Statement Option](#)” on page 215.



# Optimizing Your SQL Usage

---

<i>Overview: Optimizing Your SQL Usage</i>	55
<i>Tracing and Evaluating SQL Generation</i>	56
<i>About the DBIDIRECTEXEC System Option</i>	56
<i>Passing Functions to the DBMS Using PROC SQL</i>	58
Overview of Passing Functions to a DBMS	58
Using SQL_FUNCTIONS=ALL	59
Functions Where Results Might Vary: MOD Function	59
Processing the DISTINCT Operator	60
<i>Date Arithmetic with Queries</i>	61
<i>Passing Joins to the DBMS</i>	61
Overview of Passing Joins to the DBMS	61
Join Examples	62
Passing Joins That Use Two or More LIBNAME Statements	62
<i>When Passing Joins to the DBMS Fails</i>	64
<i>Using the DIRECT_SQL= LIBNAME Option</i>	65
<i>Passing the WHERE Clause to the DBMS</i>	66
General Guidelines for WHERE Clauses	66
Passing Functions to the DBMS Using WHERE Clauses	67
<i>Advanced PROC SQL Optimization Hints</i>	67
Overview	67
MULTI_DATASRC_OPT=	68

---

## Overview: Optimizing Your SQL Usage

SAS/ACCESS takes advantage of DBMS capabilities by passing certain SQL operations to the DBMS whenever possible. This can reduce data movement, which can improve performance. The performance impact can be significant when you access large DBMS tables and the SQL that is passed to the DBMS subsets the

table to reduce the number of rows. SAS/ACCESS sends operations to the DBMS for processing in these situations.

- When you use the [SQL pass-through facility](#), you submit DBMS-specific SQL statements that are sent directly to the DBMS for execution. For example, when you submit Transact-SQL statements to be passed to an SAP ASE database.
- When SAS/ACCESS can translate operations into the SQL of the DBMS, processing is done in the DBMS. When you use the [SAS/ACCESS LIBNAME statement](#), you submit SAS statements that SAS/ACCESS can often translate into the SQL of the DBMS and then pass to the DBMS for processing.

With the automatic translation abilities of SAS/ACCESS, you can often achieve the performance benefits of the SQL pass-through facility without needing to write DBMS-specific SQL code. The following sections describe the SAS SQL operations that SAS/ACCESS can pass to the DBMS for processing. For information about passing WHERE clauses to the DBMS, see “[Passing the WHERE Clause to the DBMS](#)”.

Typically, the SQL that SAS generates for the DBMS is highly optimized. However, in cases when SQL generation is partial or not optimal, use the information in this chapter to modify the generated SQL that is created by SAS.

## Tracing and Evaluating SQL Generation

SAS provides a tracing facility that you can use to review the SQL code that SAS generates and passes to the DBMS. Specify the `SASTRACE=` option, as shown in this example for Netezza, to display the generated SQL to the SAS log.

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix msglevel=i;
libname mydb netezza server='myserver.com' database=mydbms user=myuser
password=mypwd;

proc sql;
  select count(*) from mydb.class;
quit;
```

This code generates a query to a Netezza database:

```
select COUNT(*) from MYDBMS..class TXT_1
```

For more information, see “[SASTRACE= SAS System Option](#)” on page 669.

## About the DBIDIRECTEXEC System Option

When the [DBIDIRECTEXEC system option](#) is enabled, PROC SQL generates database-specific queries for CREATE TABLE AS SELECT, INSERT INTO TABLE ...

VALUES, INSERT INTO TABLE ... SELECT, and DELETE FROM TABLE statements.  
This option can greatly improve PROC SQL query performance.

---

**Note:** The DBIDIRECTEXEC system option applies only to queries that are generated by using PROC SQL. DBIDIRECTEXEC has no impact on code that is passed to a database from the DATA step.

---

This example uses the SASHELP.CLASS data set and shows how PROC SQL generates a database-specific CREATE TABLE AS SELECT query.

```
options sastrace=',,d' sastraceloc=saslog nostsuffix msglevel=i;
options dbidirectexec;

libname mynetdb netezza server='myserver.com' database=mydb user=myuser
      password=mypwd;

/* clean up */
proc delete data=mynetdb.class; run;
proc delete data=mynetdb.class2; run;

/* create test data */
data mynetdb.class;
  set sashelp.class;
run;

proc sql;
  create table mynetdb.class2 as select * from mynetdb.class;
quit;
```

Here is the generated SQL in the SAS log.

```
CREATE TABLE MYDB..class2 as ( select TXT_1."NAME", TXT_1."SEX",
      TXT_1."AGE", TXT_1."HEIGHT", TXT_1."WEIGHT" from MYDB..class TXT_1 )
```

When NODBIDIRECTEXEC is specified (DBIDIRECTEXEC is disabled), the CREATE TABLE AS SELECT operation is decomposed into separate CREATE, SELECT, and INSERT operations.

Here is the resulting SQL code for the same program when NODBIDIRECTEXEC is specified.

```
SELECT * FROM MYDB..class

CREATE TABLE MYDB..class2 (NAME VARCHAR(8),SEX VARCHAR(1),AGE DOUBLE,
      HEIGHT DOUBLE,WEIGHT DOUBLE)

NETEZZA_21: Prepared: on connection 3
INSERT INTO MYDB..class2 (NAME,SEX,AGE,HEIGHT,WEIGHT) VALUES ( ? , ? ,
      ? , ? , ? )
```

If a query fails when DBIDIRECTEXEC is enabled, SAS retries the query as if NODBIDIRECTEXEC were specified.

# Passing Functions to the DBMS Using PROC SQL

## Overview of Passing Functions to a DBMS

In most cases, SAS/ACCESS automatically passes down SAS SQL aggregate functions, such as MIN, MAX, AVG, MEAN, FREQ, N, SUM, and COUNT, to the DBMS. These are SQL ANSI-defined aggregate functions. They typically automatically pass down functions that have the same behavior.

Here is an example that shows how the AVG function is passed to Netezza.

```
libname x netezza database=TEST ...;
/* Create test data */
proc delete data=x.class;
run;

data x.class;
  set sashelp.class;
run;

/* Call aggregate function AVG */
proc sql;
  select avg(age)
  from x.class;
quit;
```

PROC SQL generates this DBMS query:

```
select AVG(TXT_1."AGE") from TEST..class TXT_1
```

In some cases, such as when remerging data is required, only the SELECT statement is passed to the database. Additional calculations, including aggregate functions, are completed by SAS. In these cases, a note is printed to the SAS log when SASTRACE= is enabled.

PROC SQL also generates DBMS-specific queries when functions are present in the WHERE clause. Here is an example that shows how SAS/ACCESS translates the SAS UPCASE function into the Netezza UPPER function.

```
proc sql;
  select avg(age)
  from x.class
  where upcase(sex) = 'F';
quit;
```

PROC SQL generates this DBMS query:

```
NETEZZA_15: Prepared: on connection 3
select AVG(TXT_1."AGE") from TEST..class TXT_1 where upper(TXT_1."SEX") = 'F'
```

The functions that are passed to the DBMS vary based on the specific SAS/ACCESS product. See the “Passing SAS Functions” section for your SAS/ACCESS interface for a list of SAS functions and the DBMS functions that correspond to each one.

## Using SQL\_FUNCTIONS=ALL

By default, the DBMS functions that are called in a query that is generated by SAS produce the same results as the comparable function in SAS. However, there are some DBMS functions that produce slightly different results than those that a comparable SAS function generates. Because different results are produced, SAS does not call those functions by default. Sometimes the results that are generated by a DBMS function are very close to the results that would be generated in SAS. In those cases, it is preferable to pass the query to the DBMS and to call the DBMS function. Evaluate the differences in results for your data to make sure that results are acceptable for your needs.

To enable use of these functions, specify the SQL\_FUNCTIONS=ALL LIBNAME option. See the documentation for your interface about passing SAS functions to determine which functions are passed down when you specify SQL\_FUNCTIONS=ALL. Alternatively, you can generate the list of options that are passed to your DBMS by using the SQL\_FUNCTIONS\_COPY= LIBNAME option. Here is an example for a Netezza database:

```
/* generate data set WORK.SF, contains default SQL functions */
libname x netezza server='myserver.com' database=mydb user=myuser
password=mypwd
SQL_FUNCTIONS_COPY=WORK.SF ;

/* generate data set WORK.SFALL, contains ALL target SQL functions */
libname x netezza server='myserver.com' database=mydb user=myuser
password=mypwd
SQL_FUNCTIONS=ALL SQL_FUNCTIONS_COPY=WORK.SFALL ;

/* use PROC SQL to create SFDIFF containing just the non-default
functions */
proc sql;
create table sfdiff as
select * from SFALL
except all
select * from SF;
quit;
```

It is also possible to add functions to the SQL dictionary for your interface. For more information, see [“SQL\\_FUNCTIONS= LIBNAME Statement Option” on page 324](#).

## Functions Where Results Might Vary: MOD Function

In general, SAS functions that are passed through to a DBMS yield the same results. That is, your results are the same whether you pass a function to SAS or to

your DBMS. However, in some cases, SAS might yield a different result than you would obtain with the same function call to your DBMS.

The MOD function is one function that might give a different result depending on where the function runs. More specifically, when you pass non-integer arguments to the MOD function, the results might differ. The MOD function returns the remainder from the division of the first argument by the second argument. In SAS, both arguments can be non-integers, and the calculations are performed without altering the arguments. In some DBMSs, such as DB2 or PostgreSQL, non-integer arguments are truncated to the nearest integer before performing the division.

As a best practice, be sure to understand how the MOD function works on your DBMS or you might get unexpected results.

## Processing the DISTINCT Operator

The DISTINCT operator specifies that only distinct values across all variables in the SELECT clause should be returned by a query. The DISTINCT operator can be inefficient when it is used to generate query results within SAS. Therefore, SAS always attempts to pass queries that use the DISTINCT operator to the DBMS.

To demonstrate the impact of using the DISTINCT operator, consider these queries.

```
libname x netezza server='myserver.com' database=mydb user=myuser
      password=mypwd;

/* The LOG function is not passed to the DBMS */
proc sql;
  select log(age) from x.class;
quit;
```

This call to PROC SQL generates this SQL query to be passed to the database.

```
SELECT "AGE" FROM TEST..class
```

Adding DISTINCT causes the LOG function to be passed to the DBMS. (For Netezza, the SAS LOG function is translated to the Netezza LN function.)

```
libname x netezza server='myserver.com' database=mydb user=myuser
      password=mypwd;

/* The LOG function is not passed to the DBMS */
proc sql;
  select distinct log(age) from x.class;
quit;
```

This call to PROC SQL generates this SQL query to be passed to the database.

```
select distinct ln(TXT_1."AGE") from TEST..class TXT_1
```

Notice that in the first query, the LN function is not called. The first query requires a full table scan. Because each row is being examined, SAS completes the LOG computation as it processes each row.

In the second query, the DISTINCT operator causes the processing to be passed to the DBMS for optimal performance. Therefore, the LOG function (translated to the LN function for Netezza) is also passed to the DBMS.

---

# Date Arithmetic with Queries

When you work with dates, be aware that date arithmetic is not passed through to your external DBMS. For example, suppose that you want to calculate a person's age as the difference between the current date and their birth date. If you include this calculation as part of a query, the data from your DBMS would be pulled first into SAS so that the calculation can be done. Therefore, it is recommended that you avoid using date arithmetic when you query an external DBMS.

---

# Passing Joins to the DBMS

---

## Overview of Passing Joins to the DBMS

When you perform a join across SAS/ACCESS librefs in a single DBMS, PROC SQL can often pass the join to the DBMS for processing. Passing the JOIN operation to the target DBMS results in less data movement and improves performance compared to performing a join in SAS. When a join is performed in SAS, the data must first be loaded into SAS before the join is completed. Standard INNER, OUTER LEFT, OUTER RIGHT, OUTER FULL, and CROSS joins can be passed to the target DBMS. For more information about using PROC SQL for joins, see “[Retrieving Data from Multiple Tables](#)” in *SAS SQL Procedure User’s Guide*.

Before PROC SQL implements a join, it checks to see whether the DBMS can process the join. A comparison is made using the [SAS/ACCESS LIBNAME statement](#) for the librefs. Certain criteria must be met for the join to proceed. See the information in the DBMS-specific reference for your interface to see the criteria that are required to pass a join to the DBMS.

If PROC SQL determines that a query meets the criteria, it passes the join to the DBMS. The DBMS then performs the join and returns only the results to SAS. PROC SQL processes the join in SAS if the query cannot be passed to the DBMS.

These types of joins are eligible for passing to the DBMS.

- For all DBMSs, inner joins between two or more tables.
- For DBMSs that support ANSI outer join syntax, outer joins between two or more DBMS tables.

To determine whether a JOIN operation is passed to the DBMS and to diagnose issues, use the SASTRACE= option. For more information, see “[Tracing and Evaluating SQL Generation](#)” on page 56.

---

## Join Examples

In this example, TABLE1 and TABLE2 are large DBMS tables. Each has a column named DeptNo. You want to retrieve the rows from an inner join of these tables. PROC SQL detects the join between two tables in the DBLIB library (which references an Oracle database), and SAS/ACCESS passes the join directly to the DBMS. The DBMS processes the inner join between the two tables and returns only the resulting rows to SAS.

```
libname dblib oracle user=myusr1 password=mypwd1;
proc sql;
    select tab1.deptno, tab1.dname
        from dblib.table1 tab1, dblib.table2 tab2
        where tab1.deptno = tab2.deptno;
quit;
```

The query is passed to the DBMS and generates this Oracle code.

```
select TXT_1."deptno", TXT_1."dname" from TABLE1 TXT_1, TABLE2 TXT_2
    where TXT_1."deptno" = TXT_2."deptno"
```

In this example, an outer join between two Oracle tables, TABLE1 and TABLE2, is passed to the DBMS for processing.

```
libname myorplib oracle user=myusr1 password=mypwd1;
proc sql;
    select * from myorplib.table1 right join myorplib.table2
        on table1.x = table2.x
        where table2.x > 1;
quit;
```

The query is passed to the DBMS and generates this Oracle code.

```
select TXT_1."X", TXT_2."X" from TABLE1 TXT_1 right join TABLE2 TXT_2 on
    TXT_1."X" = TXT_2."X" where TXT_2."X" > 1
```

---

## Passing Joins That Use Two or More LIBNAME Statements

SAS/ACCESS can pass down JOIN operations when more than one LIBNAME statement is involved. In this case, SAS/ACCESS starts by comparing the LIBNAME statements to see whether they are equivalent. Two LIBNAME connections are equivalent when they share these attributes:

- database source types
- connection options, including the user ID, server, and so on

Equivalent LIBNAME connections can connect to different database instances, as long as they are the same type of database and are on the same server. If the LIBNAME statements are determined to be equivalent, then a JOIN operation that uses both LIBNAME connections can be passed to the DBMS.

---

**Note:** When you join tables across multiple LIBNAME connections, implicit pass-through uses the first connection to process the data. LIBNAME options from subsequent connections are ignored.

---

This example demonstrates how SAS performs an inner join operation that uses two librefs, X and Y. Libref X references the Test database, and libref Y references the Test2 database.

```

options sastrace=',,d' sastraceloc=saslog nostsuffix;
libname x netezza server='<server>.com' database=test ...;
libname y netezza server='<server>.com' database=test2 ...;

/* drop tables if already in existence */
%macro drop(table);
%if %sysfunc(exist(&table)) %then proc delete data=&table.; run;
%mend drop;

%drop(x.left1);
%drop(y.right1);

/* Create DBMS tables */
data x.left1;
length lname $12 lcity $12;
input keyx lname lcity;
cards;
1 Lewis Durham
2 Cummings Raleigh
2 Kent Cary
3 Eaton Durham
;

data y.right1;
length rname $12 rcity $12;
input keyx rname rcity;
cards;
1 Johnston Durham
2 Dean Cary
2 Corrigan Raleigh
4 Gomez Cary
;

/* Inner join across different LIBNAME statements */
proc sql;
select *
from x.left1 inner join y.right1
on left1.keyx=right1.keyx;
quit;

```

This example works because the Netezza DBMS supports a cross-database JOIN operation. If the credentials for the X libref were different from the credentials for libref Y, then SAS would not consider these librefs to be equivalent and would not pass the join to the DBMS for processing.

# When Passing Joins to the DBMS Fails

There are several reasons why a join under PROC SQL might not be passed to the DBMS for processing. In general, the success of the join depends on the nature of the SQL that was coded and the DBMS's acceptance of the generated syntax. It is also greatly influenced by the use of option values. Here are the primary reasons why join statements might fail to be passed.

## **The DBMS does not accept the generated SQL syntax.**

PROC SQL attempts to pass an SQL join query directly to the DBMS for processing. The DBMS can reject the syntax for any number of reasons. In this event, PROC SQL attempts to open both tables individually and perform the join in SAS.

Each DBMS has specific limitations about what it allows in a join. Here are some situations that are not supported in all databases:

- Mixing an outer join with an inner join
- Performing a full outer join
- Using prohibited operators in an ON clause or using a literal operand in an ON clause.
- Using a WHERE clause in addition to an ON clause

For the specific criteria that apply to your DBMS, see the information about passing joins for your SAS/ACCESS interface.

## **The SQL query involves multiple librefs that do not share connection characteristics.**

If the librefs are specified using different data sources (DBMS engines), servers, user IDs, or any other connection options, PROC SQL does not attempt to pass the statement to the DBMS for direct processing.

## **Using data set options in the query**

Specifying most data set options on a table that is referenced in an SQL query prohibits SAS from successfully passing the statement to the DBMS for direct processing.

---

**Note:** The PRE\_STMT\_OPTS=, POST\_STMT\_OPTS=, PRE\_TABLE\_OPTS=, and POST\_TABLE\_OPTS= data set options are exceptions, because SAS passes the arguments for these options down to the DBMS for processing.

---

## **Using certain LIBNAME options**

Specifying LIBNAME options that request such member-level controls as table locks ([READ\\_LOCK\\_TYPE=](#) or [UPDATE\\_LOCK\\_TYPE=](#)) prohibits the statement from successfully passing to the DBMS for direct processing.

## **Using SAS functions on the SELECT clause**

Specifying SAS functions on the SELECT clause can prevent joins from being passed.

---

# Using the DIRECT\_SQL= LIBNAME Option

The DIRECT\_SQL= LIBNAME option enables you to restrict the SQL code that is passed to a database when you perform a join or when you use a WHERE clause. Use of this option is required only in special circumstances.

For example, if you have NULL values in a DBMS column that is used in a WHERE clause, your results might differ based on whether WHERE processing occurs in the DBMS or in SAS. Many DBMSs remove NULL values before processing a WHERE clause, but SAS does not remove NULL values. To prevent WHERE clauses from being passed to the DBMS, specify DIRECT\_SQL=NOWHERE. For more information, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

The default value for the [DIRECT\\_SQL= LIBNAME option](#) is YES. PROC SQL attempts to pass SQL joins directly to the DBMS for processing. Other values for DIRECT\_SQL= influence the nature of the SQL statements that PROC SQL tries to pass down to the DBMS or if it tries to pass anything at all.

## DIRECT\_SQL=NO

PROC SQL does not attempt to pass SQL join queries to the DBMS. However, other SQL statements can be passed. If the [MULTI\\_DATASRC\\_OPT=](#) is in effect, the generated SQL can also be passed.

## DIRECT\_SQL=NONE

PROC SQL does not attempt to pass any SQL directly to the DBMS for processing.

## DIRECT\_SQL=NOFUNCTIONS

PROC SQL does not pass any statement to the DBMS if that statement contains a function other than a summary function. Normally, PROC SQL attempts to pass down any functions coded in the SQL to the DBMS, provided the DBMS supports the given function.

## DIRECT\_SQL=NOGENSQL

PROC SQL does not attempt to pass SQL join queries to the DBMS. Other SQL statements can be passed down, however. If the [MULTI\\_DATASRC\\_OPT=](#) is in effect, the generated SQL can be passed.

## DIRECT\_SQL=NOMULTOUTJOINS

PROC SQL does not attempt to pass any multiple outer joins to the DBMS for direct processing. Other SQL statements can be passed, however, including portions of a multiple outer join.

## DIRECT\_SQL=NOWHERE

PROC SQL attempts to pass SQL to the DBMS, including SQL joins. However, it does not pass any WHERE clauses that are associated with the SQL statement. This causes any join that is attempted with direct processing to fail.

## DIRECT\_SQL=ONCE

PROC SQL passes generated SQL to the DBMS for processing. However, it tries only once. It does not try again if the first attempt fails.

**DIRECT\_SQL=YES**

PROC SQL automatically attempts to pass the SQL join query to the DBMS. This is the default setting for this option. The join attempt could fail because of a DBMS return code. If this happens, PROC SQL attempts to open both tables individually and perform the join in SAS.

# Passing the WHERE Clause to the DBMS

## General Guidelines for WHERE Clauses

Efficient WHERE clause processing can reduce the data movement between SAS and the DBMS, and SAS/ACCESS passes WHERE clauses to the DBMS whenever possible. When a WHERE clause is passed to the DBMS, the DBMS subsets the data, resulting in less data to be transferred between the DBMS and SAS. The performance improvement can be significant when you access large DBMS tables.

Follow the general guidelines in this table for writing efficient WHERE clauses.

**Table 6.1 Efficient WHERE Clause Guidelines**

Guideline	Inefficient	Efficient
Avoid the NOT operator if you can use an equivalent form.	Inefficient: where zipcode not > 8000	Efficient: where zipcode <= 8000
Avoid LIKE predicates that begin with % or _ .	Inefficient: where COUNTRY like '%INA'	Efficient: where COUNTRY like 'A%INA'
Avoid arithmetic expressions in a predicate.	Inefficient: where SALARY > 12 * 4000.00	Efficient: where SALARY > 48000.00

Use **DBINDEX=**, **DBKEY=**, and **MULTI\_DATASRC\_OPT=** when appropriate.

If you have NULL values in a DBMS column that is used in a WHERE clause, be aware that your results might differ depending on whether the WHERE clause is processed in SAS or is passed to the DBMS for processing. This is because DBMSs tend to remove NULL values from consideration in a WHERE clause, but SAS does not. For more information and strategies for this situation, see “[Using the DIRECT\\_SQL= LIBNAME Option](#)” on page 65.

---

## Passing Functions to the DBMS Using WHERE Clauses

When you use the SAS/ACCESS LIBNAME statement, SAS/ACCESS translates several SAS functions in WHERE clauses into DBMS-specific functions so that they can be passed to the DBMS.

In this SAS code, SAS can translate the FLOOR function into a DBMS function and pass the WHERE clause to the DBMS.

```
libname myoralib oracle user=myusr1 password=mypwd1;
proc print data=myoralib.personnel;
    where floor(hourlywage)+floor(tips)<10;
run;
```

Generated SQL that the DBMS processes would be similar to this code.

```
SELECT "HOURLYWAGE", "TIPS" FROM PERSONNEL
WHERE ((FLOOR("HOURLYWAGE") + FLOOR("TIPS")) < 10)
```

If the WHERE clause contains a function that SAS cannot translate into a DBMS function, SAS retrieves all rows from the DBMS and applies the WHERE clause.

The functions that are passed down are different for each DBMS. See the information about passing functions to the DBMS for your SAS/ACCESS interface to determine which functions SAS/ACCESS translates.

---

## Advanced PROC SQL Optimization Hints

---

### Overview

Most users never need to use the options that are described in this topic. However, there are some cases when slightly different SQL query generation can lead to better query performance. These query modifications might be dependent on your data, such as when performing a join operation that depends on the relative size of the tables that you are joining.

The options that are described in these sections are considered to be hints. That is, PROC SQL might ignore these options if it determines that the options cannot be used to pass down a query.

---

## MULTI\_DATASRC\_OPT=

The [MULTI\\_DATASRC\\_OPT= LIBNAME](#) option is a hint to the PROC SQL optimizer. Specifying MULTI\_DATASRC\_OPT=IN\_CLAUSE instructs the PROC SQL optimizer to generate an IN clause for a join of two tables. This prevents SAS from retrieving all rows from the DBMS tables. This hint is specific to PROC SQL and does not apply to DATA step processing.

This option typically improves performance when these conditions are true:

- One of the tables is small enough to be a candidate for IN clause processing. This depends on the number of unique values that are allowed for IN clause processing for your DBMS.
- The other table is significantly larger than the smaller table.
- The JOIN condition is not passed to the underlying database. For more information, see [“When Passing Joins to the DBMS Fails” on page 64](#).

Because the MULTI\_DATASRC\_OPT= LIBNAME option acts as a hint, it might be ignored by PROC SQL.

When you join two DBMS tables, MULTI\_DATASRC\_OPT= performs a row count operation on each table to determine the larger table. If you already know which table is larger, indicate that with the [DBLARGETABLE= data set option](#). When you use the DBLARGETABLE= data set option, SAS does not have to compute which table is larger.

---

**Note:** Whenever possible, SAS uses table statistics to avoid calculating the number of rows in a table. For example, SAS/ACCESS Interface to Hadoop uses Hive statistics to estimate the row count of a DBMS table when those statistics are present. If table statistics are not present, a SELECT COUNT(\*) query is executed.

---

# Threaded Reads

---

<i>Overview: Threaded Reads in SAS/ACCESS</i> .....	69
<i>Underlying Technology of Threaded Reads</i> .....	70
<i>SAS/ACCESS Interfaces and Threaded Reads</i> .....	70
<i>Scope of Threaded Reads</i> .....	71
<i>Options That Affect Threaded Reads</i> .....	71
<i>Generating Trace Information for Threaded Reads</i> .....	72
<i>Performance Impact of Threaded Reads</i> .....	75
<i>Autopartitioning Techniques in SAS/ACCESS</i> .....	76
<i>Data Ordering in SAS/ACCESS</i> .....	77
<i>Two-Pass Processing for SAS Threaded Applications</i> .....	77
<i>When Threaded Reads Do Not Occur</i> .....	78
<i>Summary of Threaded Reads</i> .....	78

---

## Overview: Threaded Reads in SAS/ACCESS

In SAS 8 and earlier, SAS/ACCESS opened a single connection to the DBMS to read a table. SAS statements requesting data were converted to an SQL statement and passed to the DBMS. The DBMS processed the SQL statement, produced a result set consisting of table rows and columns, and transferred the result set back to SAS on the single connection.

With a threaded Read, you can reduce the table read time by retrieving the result set on multiple connections between SAS and the DBMS. SAS can create multiple threads, and a read connection is established between the DBMS and each SAS thread. The result set is partitioned across the connections, and rows are passed to

SAS simultaneously (in parallel) across the connections, which improves performance.

---

## Underlying Technology of Threaded Reads

To perform a threaded Read, SAS first creates threads within the SAS session. Threads are standard operating system tasks that SAS controls. SAS then establishes a DBMS connection on each thread, causes the DBMS to partition the result set, and reads one partition per thread. To cause the partitioning, SAS appends a WHERE clause to the SQL so that a single SQL statement becomes multiple SQL statements, one for each thread. Here is an example.

```
proc reg SIMPLE  
data=dblib.salesdata (keep=salesnumber maxsales);  
  
ar _ALL_;  
run;
```

Previous versions of SAS opened a single connection and issued:

```
SELECT salesnumber,maxsales FROM SALES DATA;
```

Assuming that SalesData has an EmployeeNum integer column, SAS 9.1 might open two connections by issuing these two statements:

```
SELECT salesnumber,maxsales FROM salesdata WHERE (EMPLOYEEENUM mod 2)=0;
```

```
SELECT salesnumber,maxsales FROM SALES DATA WHERE (EMPLOYEEENUM mod 2)=1;
```

For more information about MOD, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

---

**Note:** *Might* is an important word here. Most but not all SAS/ACCESS interfaces support threaded Reads in SAS 9.1. The partitioning WHERE clauses that SAS generates vary. In cases where SAS cannot always generate partitioning WHERE clauses, the SAS user can supply them. In addition to WHERE clauses, other ways to partition data might also exist.

---

---

## SAS/ACCESS Interfaces and Threaded Reads

Threaded Reads work across all UNIX and Windows platforms where you run SAS. For more information about threaded Reads, see the DBMS-specific reference section for your specific SAS/ACCESS interface.

---

# Scope of Threaded Reads

SAS steps called threaded applications are automatically eligible for a threaded Read. Threaded applications are bottom-to-top fully threaded SAS procedures that perform data reads, numerical algorithms, and data analysis in threads. Only some SAS procedures are threaded applications. Here is a basic example of PROC REG, a SAS threaded application.

```
libname lib oracle user=myusr1 password=mypwd1;
proc reg simple
  data=lib.salesdata (keep=salesnumber maxsales);
  var _all_;
run;
```

For DBMSs, many more SAS steps can become eligible for a threaded Read, specifically, steps with a read-only table. A libref has the form Lib.DbTable, where Lib is a SAS libref that points to DBMS data, and DbTable is a DBMS table. Here are sample read-only tables for which threaded Reads can be turned on.

```
libname lib oracle user=myusr1 password=mypwd1;
proc print data=lib.dbtable;
run;

data local;
set lib.families;
where gender="F";
run;
```

An eligible SAS step can require user assistance to actually perform threaded Reads. If SAS cannot automatically generate a partitioning WHERE clause or otherwise perform threaded Reads, the user can code an option that supplies partitioning. To determine whether SAS can automatically generate a partitioning WHERE clause, use the [SASTRACE=](#) and [SASTRACELOC=](#) system options.

Threaded Reads can be turned off altogether. This eliminates additional DBMS activity associated with SAS threaded Reads, such as additional DBMS connections and multiple SQL statements.

Threaded Reads are not supported for the pass-through facility, in which you code your own DBMS-specific SQL that is passed directly to the DBMS for processing.

---

# Options That Affect Threaded Reads

For threaded Reads from DBMSs, SAS/ACCESS provides the [DBSLICE=](#) and [DBSLICEPARM=](#) data set options.

DBSLICE= applies only to a table reference. You can use it to code your own WHERE clauses to partition table data across threads, and it is useful when you are familiar with your table data. For example, if your DBMS table has a CHAR(1)

column Gender and your clients are approximately half female, Gender equally partitions the table into two parts. Here is an example.

```
proc print data=lib.dbtable (dbslice=("gender='f'" "gender='m'"));
  where dbcol>1000;
  run;
```

SAS creates two threads and about half of the data is delivered in parallel on each connection.

When you apply DBSLICEPARM=ALL instead of DBSLICE=, SAS attempts to "autopartition" the table for you. With the default DBSLICEPARM=THREADED\_APPS value, SAS automatically attempts threaded Reads only for SAS threaded applications. SAS threaded applications are SAS procedures that thread input, output, and numeric operations. DBSLICEPARM=ALL extends threaded Reads to more SAS procedures, specifically steps that read only tables. As an alternative, DBSLICEPARM=NONE turns off threaded Reads entirely. You can specify it as a data set option, a LIBNAME option, or a global SAS option.

The first argument to DBSLICEPARM= is required and extends or restricts threaded Reads. The second optional argument is not commonly used and limits the number of DBMS connections. These examples demonstrate the different uses of DBSLICEPARM=.

- UNIX or Windows SAS invocation option that turns on threaded Reads for all read-only librefs:
- ```
-dbsliceparm ALL
```
- Global SAS option that turns off threaded Reads:
- ```
option dbsliceparm=NONE;
```
- LIBNAME option that restricts threaded Reads to just SAS threaded applications:
- ```
libname lib oracle user=myusr1 password=mypwd1 dbsliceparm=THREADED_APPS;
```
- Table option that turns on threaded Reads, with a maximum of three connections in this example:

```
proc print data=lib.dbtable(dbsliceparm=(ALL,3));
  where dbcol>1000;
  run;
```

DBSLICE= and DBSLICEPARM= apply only to DBMS table reads. THREADS= and CPUCOUNT= are additional SAS system options that apply to threaded applications. For more information about these options, see [SAS System Options: Reference](#).

## Generating Trace Information for Threaded Reads

A threaded Read is a complex feature. A SAS step can be eligible for a threaded Read but not have it applied. The performance effect is not always easy to predict. Use the SASTRACE option to see whether threaded Reads occurred and to help assess performance. These examples demonstrate usage scenarios with

SAS/ACCESS to Oracle. Keep in mind that trace output is in English only and changes from release to release.

```
/*Turn on SAS tracing */
options sastrace=",t," sastraceloc=saslog nostsuffix;

/* Run a SAS job */

data work.locemp;
set trlib.MYEMPS(DBSLICEPARM=(ALL,3));
where STATE in ('GA', 'SC', 'NC') and ISTENURE=0;
run;
```

The above job produces these trace messages:

```
406  data work.locemp;
407  set trlib.MYEMPS(DBSLICEPARM=(ALL, 3));
408  where STATE in ('GA', 'SC', 'NC') and ISTENURE=0;
409  run;
```

```
ORACLE: DBSLICEPARM option set and 3 threads were requested
ORACLE: No application input on number of threads.
```

```
ORACLE: Thread 2 contains 47619 obs.
ORACLE: Thread 3 contains 47619 obs.
ORACLE: Thread 1 contains 47619 obs.
ORACLE: Threaded read enabled. Number of threads created: 3
```

If you want to see the SQL that is executed during the threaded Read, you can set tracing to `sastrace=',,t,d'` and run the job again. This time, the output contains threading information and all SQL statements that Oracle processes.

```
ORACLE_9: Prepared:
SELECT * FROM MYEMPS 418  data work.locemp;

419  set trlib.MYEMPS(DBSLICEPARM=(ALL, 3));
420  where STATE in ('GA', 'SC', 'NC') and ISTENURE=0;
421  run;

ORACLE: DBSLICEPARM option set and 3 threads were requested
ORACLE: No application input on number of threads.

ORACLE_10: Executed:
SELECT "HIREDATE", "SALARY", "GENDER", "ISTENURE",
       "STATE", "EMPNUM", "NUMCLASSES"
  FROM MYEMPS WHERE ( ( ("STATE" IN ( 'GA' , 'NC' , 'SC' ) ) ) AND
        ("ISTENURE" = 0 ) ) AND ABS(MOD("EMPNUM",3))=0

ORACLE_11: Executed:
SELECT "HIREDATE", "SALARY", "GENDER", "ISTENURE",
       "STATE", "EMPNUM", "NUMCLASSES"
  FROM MYEMPS WHERE ( ( ("STATE" IN ( 'GA' , 'NC' , 'SC' ) ) ) AND
        ("ISTENURE" = 0 ) ) AND ABS(MOD("EMPNUM",3))=1

ORACLE_12: Executed:
SELECT "HIREDATE", "SALARY", "GENDER", "ISTENURE",
       "STATE", "EMPNUM", "NUMCLASSES"
  FROM MYEMPS WHERE ( ( ("STATE" IN ( 'GA' , 'NC' , 'SC' ) ) ) AND
        ("ISTENURE" = 0 ) ) AND (ABS(MOD("EMPNUM",3))=2 OR "EMPNUM" IS NULL)
```

```

ORACLE: Thread 2 contains 47619 obs.
ORACLE: Thread 1 contains 47619 obs.
ORACLE: Thread 3 contains 47619 obs.
ORACLE: Threaded read enabled. Number of threads created: 3

```

Notice that the Oracle engine used the EMPNUM column as a partitioning column.

If a threaded Read cannot be done either because all of the candidates for autopartitioning are in the WHERE clause, or because the table does not contain a column that fits the criteria, SAS places a warning in the log. For example, the data set below uses a WHERE clause that contains all possible autopartitioning columns.

```

data work.locemp;
set trlib.MYEMPS (DBSLICEPARM=ALL);
where EMPNUM<=30 and ISENTERE=0 and SALARY<=35000 and NUMCLASSES>2;
run;

```

These warnings are displayed:

```

ORACLE: No application input on number of threads.
ORACLE: WARNING: Unable to find a partition column for use w/ MOD()
ORACLE: The engine cannot automatically generate the
partitioning WHERE clauses.
ORACLE: Using only one read connection.
ORACLE: Threading is disabled due to an error.
Application reverts to nonthreading I/O's.

```

If the SAS job contains any options that are invalid when the engine tries to perform threading, you also receive a warning.

```

libname trlib oracle user=myusr1 pw=mypwd1 path=oraserver DBSLICEPARM=(ALL);

proc print data=trlib.MYEMPS (OBS=10);
where EMPNUM<=30;
run;

```

This produces these messages:

```

ORACLE: Threading is disabled due to the
ORDER BY clause or the FIRSTOBS/OBS option.
ORACLE: Using only one read connection.

```

To produce timing information, add an S in the last slot of SASTRACE=.

```

options sastrace=',,t,s' sastraceloc=saslog nostsuffix;

data work.locemp;
set trlib.MYEMPS (DBSLICEPARM=ALL);
where EMPNUM<=10000;
run;

```

Here is the resulting timing information.

```

ORACLE: No application input on number of threads.
ORACLE: Thread 1 contains 5000 obs.
ORACLE: Thread 2 contains 5000 obs.

```

```

Thread 0 fetched 5000 rows
DBMS Threaded Read Total Time: 1234 mS
DBMS Threaded Read User CPU: 46 mS
DBMS Threaded Read System CPU: 0 mS

```

```

Thread 1 fetched 5000 rows
DBMS Threaded Read Total Time: 469 mS
DBMS Threaded Read User CPU: 15 mS
DBMS Threaded Read System CPU: 15 mS
ORACLE: Threaded read enabled. Number of threads created: 2
NOTE: There were 10000 observations read from the data set TRLIB.MYEMPS.
      WHERE EMPNUM<=10000;

Summary Statistics for ORACLE are: Total SQL prepare seconds were: 0.00167
Total seconds used by the ORACLE ACCESS engine were 7.545805

```

For more information, see the [SASTRACE= system option on page 669](#).

## Performance Impact of Threaded Reads

Threaded Reads increase performance only when the DBMS result set is large. Performance is optimal when the partitions are similar in size. Using threaded Reads should reduce the elapsed time of your SAS step, but unusual cases can slow the SAS step. They generally increase the workload on your DBMS.

For example, threaded Reads for DB2 under z/OS involve a tradeoff, generally reducing job elapsed time but increasing DB2 workload and CPU usage. See the auto partitioning documentation for DB2 under z/OS for details.

SAS automatically tries to autopartition table references for SAS in threaded applications. To determine whether autopartitioning is occurring and to assess its performance, complete these tasks.

- Turn on SAS tracing to see whether SAS is autopartitioning and to view the SQL associated with each thread.
- Know your DBMS algorithm for autopartitioning.
- Turn threaded Reads on and off, and compare the elapsed times.

Follow these guidelines to ensure optimal tuning of threaded Reads.

- Use it only when pulling large result sets into SAS from the DBMS.
- Use DBSLICE= to partition if SAS autopartitioning does not occur.
- Override autopartitioning with DBSLICE= if you can manually provide substantially better partitioning. The best partitioning equally distributes the result set across the threads.
- See the DBMS-specific reference section in this document for information and tips for your DBMS.

Threaded Reads are most effective on new, faster computer hardware running SAS, and with a powerful parallel edition of the DBMS. For example, if SAS runs on a fast single processor or on a multiprocessor machine and your DBMS runs on a high-end SMP server, you can experience substantial performance gains. However, you can experience minimal gains or even performance degradation when running SAS on an old desktop model with a nonparallel DBMS edition running on old hardware.

---

# Autopartitioning Techniques in SAS/ACCESS

SAS/ACCESS products share an autopartitioning scheme based on the MOD function. Some products support additional techniques. For example, if your Oracle tables are physically partitioned in the DBMS, SAS/ACCESS Interface to Oracle automatically partitions in accordance with Oracle physical partitions rather than using MOD. SAS/ACCESS Interface to Teradata uses FastExport or the TPT Export Operator, if available, which lets the Teradata client software FastExport Utility direct partitioning.

MOD is a mathematical function that produces the remainder of a division operation. Your DBMS table must contain a column to which SAS can apply the MOD function —a numeric column constrained to integral values. DBMS integer and small integer columns suit this purpose. Integral decimal (numeric) type columns can work as well. On each thread, SAS appends a WHERE clause to your SQL that uses the MOD function with the numeric column to create a subset of the result set. Combined, these subsets add up to exactly the result set for your original single SQL statement.

For example, assume that your original SQL that SAS produces is `SELECT CHR1, CHR2 FROM DBTAB` and that the Dtab table contains the IntCol integer column. SAS creates two threads and issues these two statements:

```
SELECT CHR1, CHR2 FROM DBTAB WHERE (MOD(INTCOL, 2)=0)
```

```
SELECT CHR1, CHR2 FROM DBTAB WHERE (MOD(INTCOL, 2)=1)
```

The first thread retrieves rows with an even value for IntCol, and the second thread retrieves rows with an odd value for IntCol. Distribution of rows across the two threads is optimal if IntCol has a 50/50 distribution of even and odd values.

SAS modifies the SQL for columns containing negative integers, for nullable columns, and to combine SAS WHERE clauses with the partitioning WHERE clauses. SAS can also run more than two threads. You use the second parameter of the DBSLICEPARM= option to increase the number of threads.

The success of this technique depends on the distribution of the values in the chosen integral column. Without knowledge of the distribution, your SAS/ACCESS product attempts to select the best possible column. For example, indexed columns are given preference for some DBMSs. However, column selection is more or less a guess, and the SAS guess might cause poor distribution of the result set across the threads. If no suitable numeric column is found, MOD cannot be used at all, and threaded Reads do not occur if your SAS/ACCESS product has no other partitioning technique. For these reasons, you should explore autopartitioning particulars for your DBMS and judiciously use DBSLICE= to augment autopartitioning. See the information for your data source for specific autopartitioning details.

- [Aster on page 738](#)
- [DB2 under UNIX and PC Hosts on page 761](#)
- [DB2 under z/OS on page 797](#)

- [Greenplum on page 880](#)
- [HAWQ on page 936](#)
- [Informix on page 980](#)
- [ODBC on page 1087](#)
- [Oracle on page 1145 \(not supported under z/OS\)](#)
- [SAP ASE on page 1204](#)
- [SAP HANA on page 1236](#)
- [SAP IQ on page 1260](#)
- Teradata: For platform-specific details and special considerations, see your Teradata documentation and also [“Autopartitioning Scheme for Teradata \(Legacy\)” on page 1350](#).

---

**Note:** Beginning in SAS 9.4M8, the following interfaces are no longer available: Aster, HAWQ, MySQL on AIX, Oracle on z/OS, and Teradata on z/OS. These interfaces remain in the documentation for users who have not upgraded to SAS 9.4M8 or later.

---

---

## Data Ordering in SAS/ACCESS

The order in which table rows are delivered to SAS varies each time a step is rerun with threaded Reads. Most DBMS editions, especially increasingly popular parallel editions, do not guarantee consistent ordering.

---

## Two-Pass Processing for SAS Threaded Applications

Two-pass processing occurs when a request is received for data to be made available for multiple pass reading (that is, more than one pass through the data set). In the context of DBMS engines, this requires that as the data is read from the database, temporary spool files are written containing the read data. There is one temporary spool file per thread, and each spool file contains all data read on that thread. If three threads are specified for threaded Reads, three temporary spool files are written.

As the application requests subsequent passes of data, data is read from the temporary spool files, not reread from the database. The temporary spool files can be written on different disks. This reduces any disk read contention and enhances performance. To accomplish this, the SAS option UTILLOC= is used to specify different disk devices and directory paths when creating temporary spool files. There are several ways to specify this option.

- In the SAS config file, add this line:

```
-utilloc("C:\path" "D:\path" "E:\path")
```

- Specify the UTILLOC= option on the SAS command line:

```
/* on Windows */
sas -utilloc(c:\path d:\path e:\path)
```

```
/* on UNIX */
sas -utilloc '(\path \path2 \path3)'
```

For more information about the UTILLOC= SAS system option, see the [SAS System Options: Reference](#).

---

## When Threaded Reads Do Not Occur

Threading does not occur under these circumstances:

- when a BY statement is used in a PROC or DATA step
- when the OBS option or the FIRSTOBS option is in a PROC or DATA step
- when the KEY option or the DBKEY option is used PROC or DATA step
- if no column in the table exists to which SAS can apply the MOD function (for more information, see [“Autopartitioning Techniques in SAS/ACCESS” on page 76](#))
- if all columns within a table to which SAS can apply the MOD function are in WHERE clauses (for more information, see [“Autopartitioning Techniques in SAS/ACCESS” on page 76](#))
- if the NOTHREADS system option is specified
- if DBSLICEPARM=NONE

---

## Summary of Threaded Reads

For large reads of table data, SAS threaded Reads can speed up SAS jobs. They are particularly useful when you understand the autopartitioning technique specific to your DBMS and use DBSLICE= to manually partition only when appropriate. Look for enhancements in future SAS releases.

# National Language Support

---

|                                                |    |
|------------------------------------------------|----|
| <i>Overview: NLS for SAS/ACCESS</i> .....      | 79 |
| <i>LIBNAME and Other Options for NLS</i> ..... | 79 |
| <i>Data Types for NLS</i> .....                | 80 |

---

## Overview: NLS for SAS/ACCESS

National Language Support (NLS) is a set of features that enable a software product to function properly in every global market for which the product is targeted. SAS/ACCESS interfaces provide NLS through LIBNAME and SQL pass-through (connection) options and also data types.

---

**Note:** It is recommended that your SAS session encoding and your DBMS encoding be the same.

---

For NLS limitations that are specific to Hive, see “[Naming Conventions for SAS and Hive](#)”.

For more NLS information, see these titles.

- [SAS National Language Support \(NLS\): Reference Guide](#)
- [“Multilingual Computing with SAS 9.4” \(SAS Institute white paper\)](#)

---

## LIBNAME and Other Options for NLS

These LIBNAME and SQL pass-through (connection) options allow for byte and character conversion and length calculation.

- [ADJUST\\_BYTE\\_SEMANTIC\\_COLUMN\\_LENGTHS= LIBNAME option](#)
- [ADJUST\\_NCHAR\\_COLUMN\\_LENGTHS= LIBNAME option](#)
- [DBCLIENT\\_ENCODING\\_FIXED= LIBNAME option](#)
- [DB\\_LENGTH\\_SEMATICS\\_BYTEx LIBNAME option](#)
- [DBCLIENT\\_MAX\\_BYTES= LIBNAME option](#)
- [DBSERVER\\_ENCODING\\_FIXED= LIBNAME option](#)
- [DBSERVER\\_MAX\\_BYTES= LIBNAME option](#)

The [TRANSCODE\\_FAIL= LIBNAME](#) and [data set option](#) for SAS/ACCESS Interface to Hadoop let you specify how to handle processing and notification of transcoding errors.

The [SAS\\_REDSHIFT\\_UPDATE\\_WARINING=](#) and [SAS\\_POSTGRES\\_UPDATE\\_WARNING=](#) environment variables enable you to work with data that contains NLS characters that would otherwise trigger errors in the SAS log. When these variables are enabled, warnings are printed to the SAS log and processing continues. For more information, see [“Working with NLS Characters in Amazon Redshift Data”](#) or [“Working with NLS Characters in PostgreSQL Data”](#) on page 1189.

---

## Data Types for NLS

These data types allow for more flexible adjustment of column lengths.

- [BLOB](#)
- [CHAR](#)
- [CLOB](#)
- [DBCLOB](#)
- [NCHAR](#)
- [VARCHAR](#)

# How SAS/ACCESS Works

---

|                                                                |           |
|----------------------------------------------------------------|-----------|
| <b><i>Introduction to How SAS/ACCESS Works</i></b> .....       | <b>81</b> |
| Installation Requirements .....                                | 81        |
| SAS/ACCESS Interfaces .....                                    | 82        |
| <b><i>How the SAS/ACCESS LIBNAME Statement Works</i></b> ..... | <b>82</b> |
| Accessing Data from a DBMS Object .....                        | 82        |
| Processing Queries, Joins, and Data Functions .....            | 83        |
| <b><i>How the SQL Pass-Through Facility Works</i></b> .....    | <b>83</b> |
| <b><i>How the ACCESS Procedure Works</i></b> .....             | <b>84</b> |
| Overview: ACCESS Procedure .....                               | 84        |
| Reading Data .....                                             | 85        |
| Updating Data .....                                            | 86        |
| <b><i>How the DBLOAD Procedure Works</i></b> .....             | <b>86</b> |

---

## Introduction to How SAS/ACCESS Works

### Installation Requirements

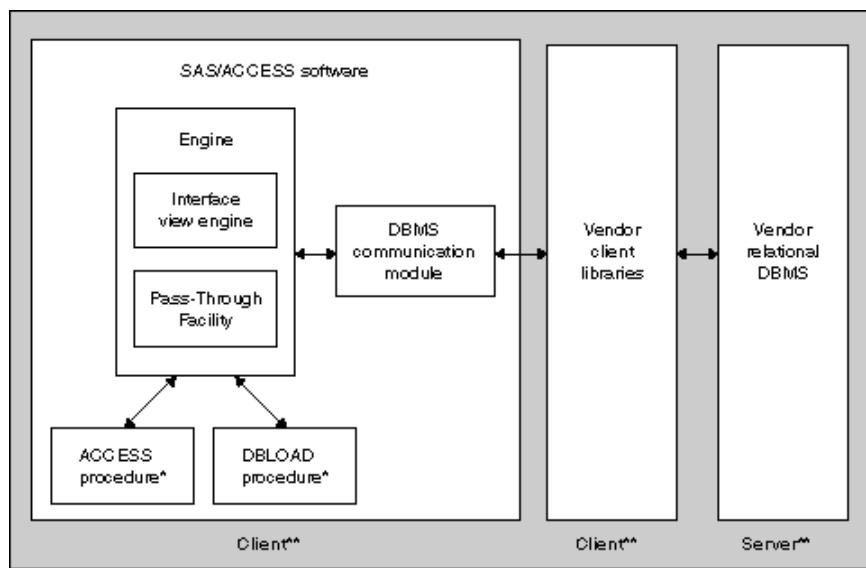
Before you use any SAS/ACCESS features, you must install Base SAS, the SAS/ACCESS interface for the DBMS that you are accessing, and any required DBMS client software. See SAS installation instructions and DBMS client installation instructions for more information.

Not all SAS/ACCESS interfaces support all features. See the DBMS-specific reference section for your SAS/ACCESS interface to determine which features are supported in your environment.

## SAS/ACCESS Interfaces

Each SAS/ACCESS interface consists of one or more data access engines that translate Read and Write requests from SAS into appropriate calls for a specific DBMS. The following image depicts the relationship between a SAS/ACCESS interface and a relational DBMS.

*Figure 9.1 How SAS Connects to the DBMS*



\* The ACCESS procedure and the DBLOAD procedure are not supported by all SAS/ACCESS interfaces.

\*\* In some cases, both client and server software can reside on the same machine.

You can call a SAS/ACCESS relational DBMS interface by using either a [LIBNAME statement](#) or a [PROC SQL statement](#). (Although you can also use [ACCESS](#) and [DBLOAD](#) procedures with some of the SAS/ACCESS relational interfaces, these procedures are no longer the recommended way to access relational database data.)

## How the SAS/ACCESS LIBNAME Statement Works

### Accessing Data from a DBMS Object

You can use SAS/ACCESS to read, update, insert, and delete data from a DBMS object as if it were a SAS data set. Here are the steps.

- 1 Start a SAS/ACCESS interface by specifying a DBMS engine name and the appropriate connection options in a LIBNAME statement.
- 2 Enter SAS requests as you would when accessing a SAS data set.
- 3 SAS/ACCESS generates DBMS-specific SQL statements that are equivalent to the SAS requests that you enter.
- 4 SAS/ACCESS submits the generated SQL to the DBMS.

The SAS/ACCESS engine specifies which operations are supported on a table and calls code that translates database operations such as open, get, put, or delete into DBMS-specific SQL syntax. SAS/ACCESS engines use an established set of routines with calls that are customized for the DBMS.

---

## Processing Queries, Joins, and Data Functions

To enhance performance, SAS/ACCESS can transparently pass queries, joins, and data functions to the DBMS for processing instead of retrieving the data from the DBMS and processing it in SAS. For example, an important use of this feature is the handling of PROC SQL queries that access DBMS data. Here is how it works.

- 1 PROC SQL examines each query to determine whether to send all or part of the query to the DBMS for processing.
- 2 A special query textualizer in PROC SQL translates queries (or query fragments) into DBMS-specific SQL syntax.
- 3 The query textualizer submits the translated query to the SAS/ACCESS engine for approval.
- 4 If SAS/ACCESS approves the translation, it sends an approval message to PROC SQL. The DBMS processes the query or query fragment and returns the results to SAS. Any queries or query fragments that cannot be passed to the DBMS are processed in SAS.

For details about tasks that SAS/ACCESS can pass to the DBMS, see the DBMS-specific reference section for your SAS/ACCESS interface.

---

## How the SQL Pass-Through Facility Works

When you read and update DBMS data with the SQL pass-through facility, SAS/ACCESS passes SQL statements directly to the DBMS for processing. Here are the steps.

- 1 Invoke PROC SQL and submit a PROC SQL CONNECT statement that includes a DBMS name and the appropriate connection options to establish a connection with a specified database.

- 2 Use a CONNECTION TO component in a PROC SQL SELECT statement to read data from a DBMS table or view.

In the SELECT statement (PROC SQL query) that you write, use the SQL that is native to your DBMS. SAS/ACCESS passes the SQL statements directly to the DBMS for processing. If the SQL syntax that you enter is correct, the DBMS processes the statement and returns any results to SAS. If the DBMS does not recognize the syntax that you enter, it returns an error that appears in the SAS log. The SELECT statement can be stored as a PROC SQL view. Here is an example.

```
proc sql;
connect to oracle (user=myusr1 password=mypwd1);
create view budget2000 as select amount_b,amount_s
from connection to oracle
(select Budgeted, Spent from annual_budget);
quit;
```

- 3 Use a PROC SQL EXECUTE statement to pass any dynamic, nonquery SQL statements (such as INSERT, DELETE, and UPDATE) to the database.

As with the CONNECTION TO component, all EXECUTE statements are passed to the DBMS exactly as you submit them. INSERT statements must contain literal values. Here is an example.

```
proc sql;
connect to oracle(user=myusr1 password=mypwd1);
execute (create view whotookorders as select ordernum, takenby,
firstname, lastname, phone from orders, employees
where orders.takenby=employees.empid) by oracle;
execute (grant select on whotookorders to myusr1) by oracle;

disconnect from oracle;
quit;
```

- 4 End the connection with the DISCONNECT statement.

For more details, see the [SQL pass-through facility on page 690](#).

## How the ACCESS Procedure Works

### Overview: ACCESS Procedure

When you use the ACCESS procedure to create an access descriptor, the SAS/ACCESS view engine requests the DBMS to execute an SQL SELECT statement to the data dictionary tables in your DBMS dynamically by using DBMS-specific call routines or interface software. The ACCESS procedure then issues the equivalent of a DESCRIBE statement to gather information about the columns in the specified table. Access descriptor information about the table and its columns is then copied into the view descriptor when it is created. It is therefore not necessary for SAS to call the DBMS when it creates a view descriptor. Here is the process.

**Note:** SAS still supports this legacy procedure, but to access your DBMS data more directly the best practice is to use the LIBNAME statement for your interface or the SQL pass-through facility.

- 1 When you provide connection information to PROC ACCESS, the SAS/ACCESS interface calls the DBMS to connect to the database.
- 2 SAS constructs a SELECT \* FROM *table-name* statement and passes it to the DBMS to retrieve information about the table from the DBMS data dictionary. This SELECT statement is based on the information that you provided to PROC ACCESS. It lets SAS determine whether the table exists and can be accessed.
- 3 The SAS/ACCESS interface calls the DBMS to obtain table description information, such as the column names, data types (including width, precision, and scale), and whether the columns accept null values.
- 4 SAS closes the connection with the DBMS.

## Reading Data

When you use a view descriptor in a DATA step or procedure to read DBMS data, the SAS/ACCESS interface view engine requests the DBMS to execute an SQL SELECT statement. Here are the steps that the interface view engine follows.

- 1 Using the connection information that is in the created view descriptor, the SAS/ACCESS interface calls the DBMS to connect to the database.
- 2 SAS constructs a SELECT statement that is based on the information stored in the view descriptor and passes this information to the DBMS. A view descriptor includes a table name and selected columns and their characteristics.
- 3 SAS retrieves the data from the DBMS table and passes it back to the SAS procedures as if it were observations in a SAS data set.
- 4 SAS closes the connection with the DBMS.

For example, if you run the following SAS program using a view descriptor, the previous steps are executed once for the PRINT procedure and a second time for the GCHART procedure. (The data that is used for the two procedures is not necessarily the same because another user might have updated the table might have been updated between procedure executions.)

```
proc print data=vlib.allemp;
run;

proc gchart data=vlib.allemp;
  vbar jobcode;
run;
```

---

## Updating Data

Use a view descriptor, DATA step, or procedure to update DBMS data similarly to when you read in data. Any of these steps might also occur.

- 1 Using the connection information that is contained in the specified access descriptor, the SAS/ACCESS interface calls the DBMS to connect to the database.
- 2 When rows are added to a table, SAS constructs an SQL INSERT statement and passes it to the DBMS. When you reference a view descriptor, use the ADD command in FSEDIT and FSVIEW, the APPEND procedure, or an INSERT statement in PROC SQL to add data to a DBMS table. (You can also use the EXECUTE statement for the SQL pass-through facility to add, delete, or modify DBMS data directly. You must use literal values when you insert data with the SQL pass-through facility.)
- 3 When rows are deleted from a DBMS table, SAS constructs an SQL DELETE statement and passes it to the DBMS. When you reference a view descriptor, you can use the DELETE command in FSEDIT and FSVIEW or a DELETE statement in PROC SQL to delete rows from a DBMS table.
- 4 When data in the rows is modified, SAS constructs an SQL UPDATE statement and passes it to the DBMS. When you reference a view descriptor, you can use FSEDIT, the MODIFY command in FSVIEW, or an INSERT statement in PROC SQL to update data in a DBMS table. You can also reference a view descriptor in the UPDATE, MODIFY, and REPLACE statements for the DATA step.
- 5 SAS closes the connection with the DBMS.

---

## How the DBLOAD Procedure Works

When you use the DBLOAD procedure to create a DBMS table, the procedure issues dynamic SQL statements to create the table and insert data from a SAS data set, DATA step view, PROC SQL view, or view descriptor into the table. Here are the steps that the SAS/ACCESS interface view engine completes.

**Note:** SAS still supports this legacy procedure, but to access your DBMS data more directly the best practice is to use the LIBNAME statement for your interface or the SQL pass-through facility.

- 1 When you supply the connection information to PROC DBLOAD, the SAS/ACCESS interface calls the DBMS to connect to the database.
- 2 SAS uses the information that the DBLOAD procedure provides to construct a SELECT \* FROM *table-name* statement and passes the information to the DBMS to determine whether the table already exists. PROC DBLOAD continues

only if a table with that name does not exist unless you use the DBLOAD APPEND option.

- 3 SAS uses the information that the DBLOAD procedure provides to construct an SQL CREATE TABLE statement and passes it to the DBMS.
- 4 SAS constructs an SQL INSERT statement for the current observation and passes it to the DBMS. New INSERT statements are constructed and executed repeatedly until all observations from the input SAS data set are passed to the DBMS. Some DBMSs have a bulk-copy capability so that a group of observations can be inserted at once. See your DBMS documentation to determine whether your DBMS has this capability.
- 5 Additional nonquery SQL statements that are specified in the DBLOAD procedure are executed as the user submitted them. The DBMS returns an error message if a statement does not execute successfully.
- 6 SAS closes the connection with the DBMS.



## 10

# In-Database Processing with SAS/ACCESS

---

*Overview: In-Database Processing* ..... 89

---

## Overview: In-Database Processing

When you use conventional processing to access data that is inside a DBMS, SAS asks the SAS/ACCESS engine for all table rows of the table that is being processed. The SAS/ACCESS engine generates an SQL SELECT \* statement, which is passed to the DBMS. That SELECT statement fetches all rows in the table, and the SAS/ACCESS engine returns them to SAS. The number of rows in the table grows over time, so network latency grows because the amount of data that is fetched from the DBMS to SAS increases.

SAS In-Database processing integrates SAS solutions, SAS analytic processes, and third-party DBMSs. Using SAS In-Database processing, you can run scoring models, Base SAS and SAS/STAT procedures, and formatted SQL queries inside the database. For more information, see *SAS and SAS Viya In-Database Products: Administrator's Guide* and *SAS In-Database Products: User's Guide*.

The following table lists the SAS products that are needed to use these in-database features.

**Table 10.1** SAS Products That Are Required for In-Database Processing

| In-Database Feature                          | Software Required                                                                        | Supported Data Providers                                     |
|----------------------------------------------|------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| format publishing and the SAS_PUT() function | Base SAS<br>SAS/ACCESS interface to the data source<br>SAS Embedded Process (Aster only) | Aster <sup>1</sup><br>DB2 under UNIX<br>Greenplum<br>Netezza |

| In-Database Feature                                                                                                                               | Software Required                                                                                                                                                                                                                                                                                                                 | Supported Data Providers                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                   | Teradata <sup>1</sup>                                                                                                                                         |
| scoring models (SAS Scoring Accelerator)                                                                                                          | Base SAS<br>SAS/ACCESS interface to the data source<br>SAS Scoring Accelerator<br>SAS Embedded Process<br>SAS Enterprise Miner<br>SAS Factory Miner (analytic store scoring) <sup>2</sup><br>SAS Model Manager (optional)                                                                                                         | Aster <sup>1</sup><br>DB2 under UNIX<br>Greenplum<br>Hadoop<br>Netezza<br>Oracle<br>SAP HANA<br>Teradata <sup>1</sup>                                         |
| scoring models (using CAS actions or PROC SCOREACCEL)                                                                                             | SAS Viya<br>Base SAS (optional)<br>SAS/ACCESS Interface to the data source<br>SAS In-Database Technologies for the data source (on SAS Viya)<br>SAS Embedded Process<br>SAS Enterprise Miner (optional)<br>SAS Factory Miner (optional)<br>SAS Visual Data Mining and Machine Learning (optional)<br>SAS Model Manager (optional) | Hadoop<br>Teradata <sup>1</sup>                                                                                                                               |
| scoring text analytics models                                                                                                                     | Base SAS<br>SAS/ACCESS Interface to Hadoop<br>SAS In-Database Code Accelerator for Hadoop<br>SAS Contextual Analysis<br>SAS Contextual Analysis In-Database Scoring for Hadoop<br>SAS Embedded Process                                                                                                                            | Hadoop                                                                                                                                                        |
| Base SAS procedures:<br>COPY <sup>3</sup><br>FREQ<br>MEANS<br>RANK <sup>4, 6, 8</sup><br>REPORT<br>SORT <sup>5, 6, 8</sup><br>SUMMARY<br>TABULATE | Base SAS<br>SAS/ACCESS interface to the data source<br>SAS In-Database Code Accelerator (PROC TRANSPOSE only)<br>SAS Embedded Process (PROC TRANSPOSE only)                                                                                                                                                                       | Amazon Redshift<br>Aster <sup>1</sup><br>DB2 (UNIX and z/OS)<br>Google BigQuery<br>Greenplum<br>Hadoop<br>HAWQ <sup>1</sup><br>Impala<br>Microsoft SQL Server |

| In-Database Feature                                            | Software Required                                                                                                                                                                             | Supported Data Providers                                                                                                                                  |
|----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| TRANSPOSE <sup>7</sup>                                         |                                                                                                                                                                                               | MySQL <sup>1,9</sup><br>Netezza<br>Oracle <sup>1</sup><br>PostgreSQL<br>SAP HANA<br>Snowflake<br>Spark<br>Teradata <sup>1</sup><br>Vertica<br>Yellowbrick |
| SAS/STAT procedures:                                           |                                                                                                                                                                                               |                                                                                                                                                           |
| CORR                                                           | Base SAS (for CORR)                                                                                                                                                                           | Teradata <sup>1</sup>                                                                                                                                     |
| CANCORR                                                        | SAS/ACCESS Interface to Teradata                                                                                                                                                              |                                                                                                                                                           |
| DMDB                                                           | SAS/STAT (for CANCORR, FACTOR, PRINCOMP, REG, SCORE, VARCLUS)                                                                                                                                 |                                                                                                                                                           |
| DMINE                                                          | SAS/ETS (for TIMESERIES)                                                                                                                                                                      |                                                                                                                                                           |
| DMREG                                                          | SAS Enterprise Miner (for DMDB, DMINE, DMREG)                                                                                                                                                 |                                                                                                                                                           |
| FACTOR                                                         | SAS Analytics Accelerator                                                                                                                                                                     |                                                                                                                                                           |
| PRINCOMP                                                       |                                                                                                                                                                                               |                                                                                                                                                           |
| REG                                                            |                                                                                                                                                                                               |                                                                                                                                                           |
| SCORE                                                          |                                                                                                                                                                                               |                                                                                                                                                           |
| TIMESERIES                                                     |                                                                                                                                                                                               |                                                                                                                                                           |
| VARCLUS                                                        |                                                                                                                                                                                               |                                                                                                                                                           |
| DS2 threaded programs (using SAS In-Database Code Accelerator) | Base SAS<br>SAS/ACCESS interface to the data source<br>SAS In-Database Code Accelerator<br>SAS Embedded Process                                                                               | Greenplum<br>Hadoop<br>Teradata <sup>1</sup>                                                                                                              |
| DATA step scoring programs (Base SAS)                          | Base SAS<br>SAS/ACCESS Interface to Hadoop<br>SAS Embedded Process                                                                                                                            | Hadoop                                                                                                                                                    |
| data quality operations                                        | Base SAS<br>SAS/ACCESS Interface to Hadoop<br>SAS/ACCESS Interface to Teradata<br>SAS In-Database Code Accelerator<br>SAS Data Loader for Hadoop<br>SAS Data Quality Accelerator for Teradata | Hadoop<br>Teradata <sup>1</sup>                                                                                                                           |

| In-Database Feature                   | Software Required                                                                                                                                                                 | Supported Data Providers        |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
|                                       | SAS Embedded Process                                                                                                                                                              |                                 |
| extract and transform data operations | Base SAS<br>SAS/ACCESS Interface to Hadoop<br>SAS/ACCESS Interface to Teradata<br>SAS Data Loader for Hadoop<br>SAS Data Quality Accelerator for Teradata<br>SAS Embedded Process | Hadoop<br>Teradata <sup>1</sup> |

- 1 Beginning in SAS 9.4M8, the following SAS/ACCESS interfaces are no longer available: Aster, HAWQ, MySQL on AIX, Oracle on z/OS, and Teradata on z/OS. These remain in the documentation for users who have not upgraded to SAS 9.4M8.
- 2 Analytic store scoring is supported only on Hadoop, SAP HANA, and Teradata.
- 3 PostgreSQL: In-database processing for PROC COPY is supported only for PostgreSQL. For more information, see the documentation for your SAS/ACCESS interface.
- 4 Google BigQuery: In-database processing for PROC RANK requires a VAR variable that is not of type FLOAT64.
- 5 Google BigQuery: In-database processing for PROC SORT requires a BY variable that is not of type FLOAT64.
- 6 Hadoop: In-database processing of PROC RANK and PROC SORT is supported by Hadoop with Hive 0.13 and later.
- 7 Hadoop, Teradata: In-database processing of PROC TRANSPOSE is supported only on Hadoop and Teradata. Additional licensing and configuration is required.
- 8 Spark: In-database processing for PROC RANK and PROC SORT requires Spark 2.4 and later.
- 9 MySQL: PROC RANK and PROC SORT are supported for MySQL 8.0.17 and later.

**PART 2****General Reference**

|                   |                                                 |            |
|-------------------|-------------------------------------------------|------------|
| <i>Chapter 11</i> | <b>SAS/ACCESS Features by Host .....</b>        | <b>95</b>  |
| <i>Chapter 12</i> | <b>The LIBNAME Statement .....</b>              | <b>123</b> |
| <i>Chapter 13</i> | <b>Data Set Options .....</b>                   | <b>365</b> |
| <i>Chapter 14</i> | <b>Macro Variables and System Options .....</b> | <b>655</b> |
| <i>Chapter 15</i> | <b>SQL Pass-Through Facility .....</b>          | <b>689</b> |



# SAS/ACCESS Features by Host

|                                                                                      |     |
|--------------------------------------------------------------------------------------|-----|
| <i>Introduction</i> .....                                                            | 96  |
| <i>Overview of Support for SAS 9.4, SAS Viya 3.5, and Data Connectors</i> .....      | 96  |
| <i>SAS/ACCESS Interface to Amazon Redshift: Supported Features</i> .....             | 97  |
| <i>SAS/ACCESS Interface to Aster: Supported Features</i> .....                       | 98  |
| <i>SAS/ACCESS Interface to DB2 under UNIX and PC Hosts: Supported Features</i> ..... | 99  |
| <i>SAS/ACCESS Interface to DB2 under z/OS: Supported Features</i> .....              | 100 |
| <i>SAS/ACCESS Interface to Google BigQuery: Supported Features</i> .....             | 101 |
| <i>SAS/ACCESS Interface to Greenplum: Supported Features</i> .....                   | 102 |
| <i>SAS/ACCESS Interface to Hadoop: Supported Features</i> .....                      | 103 |
| <i>SAS/ACCESS Interface to HAWQ: Supported Features</i> .....                        | 104 |
| <i>SAS/ACCESS Interface to Impala: Supported Features</i> .....                      | 105 |
| <i>SAS/ACCESS Interface to Informix: Supported Features</i> .....                    | 105 |
| <i>SAS/ACCESS Interface to JDBC: Supported Features</i> .....                        | 106 |
| <i>SAS/ACCESS Interface to Microsoft SQL Server: Supported Features</i> .....        | 107 |
| <i>SAS/ACCESS Interface to MySQL: Supported Features</i> .....                       | 108 |
| <i>SAS/ACCESS Interface to Netezza: Supported Features</i> .....                     | 109 |
| <i>SAS/ACCESS Interface to ODBC: Supported Features</i> .....                        | 110 |
| <i>SAS/ACCESS Interface to OLE DB: Supported Features</i> .....                      | 111 |
| <i>SAS/ACCESS Interface to Oracle: Supported Features</i> .....                      | 112 |
| <i>SAS/ACCESS Interface to PostgreSQL: Supported Features</i> .....                  | 113 |
| <i>SAS/ACCESS Interface to SAP ASE: Supported Features</i> .....                     | 114 |
| <i>SAS/ACCESS Interface to SAP HANA: Supported Features</i> .....                    | 115 |
| <i>SAS/ACCESS Interface to SAP IQ: Supported Features</i> .....                      | 116 |
| <i>SAS/ACCESS Interface to Snowflake: Supported Features</i> .....                   | 117 |
| <i>SAS/ACCESS Interface to Spark: Supported Features</i> .....                       | 118 |

|                                                                |     |
|----------------------------------------------------------------|-----|
| <i>SAS/ACCESS Interface to Teradata: Supported Features</i>    | 118 |
| <i>SAS/ACCESS Interface to Vertica: Supported Features</i>     | 119 |
| <i>SAS/ACCESS Interface to Yellowbrick: Supported Features</i> | 120 |

---

## Introduction

This chapter provides a quick summary, by host environment, of the features that are available for your SAS/ACCESS interface.

For detailed information about your particular interface, see the SAS system requirements and configuration guide documents, which are available at [SAS Support](#).

## Overview of Support for SAS 9.4, SAS Viya 3.5, and Data Connectors

This table lists high-level support for SAS 9.4 and for SAS Viya 3.5, and lists whether a SAS/ACCESS interface has an associated data connector or data connect accelerator. For more detailed information about supported platforms, see the topic for your SAS/ACCESS interface in the remainder of this chapter.

**Table 11.1** Overview of Support for SAS 9.4, SAS Viya 3.5, and Data Connectors for Each SAS/ACCESS Interface

| SAS/ACCESS Interface        | SAS 9.4 | SAS Viya 3.5 | SAS Data Connector<br>(Serial Data Transfer) | SAS Data Connect Accelerator <sup>1</sup><br>(Parallel data transfer) |
|-----------------------------|---------|--------------|----------------------------------------------|-----------------------------------------------------------------------|
| Amazon Redshift             | •       | •            | •                                            |                                                                       |
| Aster                       | •       |              |                                              |                                                                       |
| DB2 under UNIX and PC Hosts | •       | •            | •                                            |                                                                       |
| DB2 under z/OS              | •       |              |                                              |                                                                       |
| Google BigQuery             | •       | •            | •                                            |                                                                       |
| Greenplum                   | •       | •            |                                              |                                                                       |

|                      |   |   |   |   |
|----------------------|---|---|---|---|
| Hadoop               | • | • | • | • |
| HAWQ                 | • | • |   |   |
| Impala               | • | • | • |   |
| Informix             | • |   |   |   |
| JDBC                 | • | • | • |   |
| Microsoft SQL Server | • | • | • |   |
| MySQL                | • | • | • |   |
| Netezza              | • | • |   |   |
| ODBC                 | • | • | • |   |
| OLE DB               | • |   |   |   |
| Oracle               | • | • | • |   |
| PostgreSQL           | • | • | • |   |
| SAP ASE              | • | • |   |   |
| SAP HANA             | • | • | • |   |
| SAP IQ               | • |   |   |   |
| Snowflake            | • | • | • |   |
| Spark                | • |   |   |   |
| Teradata             | • | • | • | • |
| Vertica              | • | • | • |   |
| Yellowbrick          | • |   |   |   |

1 The SAS Data Connect Accelerator is available when you license SAS In-Database Technologies with your SAS/ACCESS interface.

---

# SAS/ACCESS Interface to Amazon Redshift: Supported Features

Here are the features that SAS/ACCESS Interface to Amazon Redshift supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.2** Features by Host Environment for Amazon Redshift

| Platform                       | SAS 9.4                      |                           |                                   | SAS Viya 3.5                 |                           |                                   |
|--------------------------------|------------------------------|---------------------------|-----------------------------------|------------------------------|---------------------------|-----------------------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support |
| AIX                            | •                            | •                         | •                                 |                              |                           |                                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                                 |                              |                           |                                   |
| 64-bit Linux                   | •                            | •                         | •                                 | •                            | •                         | •                                 |
| Solaris for SPARC              | •                            | •                         | •                                 |                              |                           |                                   |
| 64-bit Solaris                 | •                            | •                         | •                                 |                              |                           |                                   |
| 64-bit Microsoft Windows       | •                            | •                         | •                                 |                              |                           |                                   |
| 32-bit Microsoft Windows       | •                            | •                         | •                                 |                              |                           |                                   |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for Amazon Redshift](#)” on page 718.

## SAS/ACCESS Interface to Aster: Supported Features

**Note:** Beginning in SAS 9.4M8, SAS/ACCESS Interface to Aster is no longer available. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

Here are the features that SAS/ACCESS Interface to Aster supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.3** Features by Host Environment for Aster

| Platform                 | SAS 9.4                      |                           |                                   |
|--------------------------|------------------------------|---------------------------|-----------------------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support |
| AIX                      | •                            | •                         | •                                 |
| 64-bit Linux             | •                            | •                         | •                                 |
| 32-bit Microsoft Windows | •                            | •                         | •                                 |
| 64-bit Microsoft Windows | •                            | •                         | •                                 |
| Solaris for SPARC        | •                            | •                         | •                                 |

**Note:** Support for the AIX and Solaris for SPARC platforms was added in [SAS 9.4M1](#).

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page [4](#) and “[Bulk Loading and Unloading for Aster](#)” on page [742](#).

---

## SAS/ACCESS Interface to DB2 under UNIX and PC Hosts: Supported Features

Here are the features that SAS/ACCESS Interface to DB2 under UNIX and PC Hosts supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.4** Features by Host Environment for DB2 under UNIX and PC Hosts

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| Solaris for SPARC              | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

If you have SAS 9.4, then SAS/ACCESS Interface to DB2 under UNIX and PC Hosts also supports the DBLOAD procedure for all platforms. The DBLOAD procedure is not included with SAS Viya 3.5.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for DB2 under UNIX and PC Hosts](#)” on page 772.

---

## SAS/ACCESS Interface to DB2 under z/OS: Supported Features

Here are the features that SAS/ACCESS Interface to DB2 under z/OS supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.5** Features by Host Environment for DB2 under z/OS

| Platform | SAS 9.4                      |                           |                   |
|----------|------------------------------|---------------------------|-------------------|
|          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| z/OS     | •                            | •                         | •                 |

SAS/ACCESS Interface to DB2 under z/OS supports the ACCESS and DBLOAD procedures.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for DB2 under z/OS](#)” on page 822.

## SAS/ACCESS Interface to Google BigQuery: Supported Features

Here are the features that SAS/ACCESS Interface to BigQuery supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.6** Features by Host Environment for BigQuery

| Platform     | SAS 9.4                      |                           |                                   | SAS Viya 3.5                 |                           |                                   |
|--------------|------------------------------|---------------------------|-----------------------------------|------------------------------|---------------------------|-----------------------------------|
|              | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support |
| 64-bit Linux | •                            | •                         | •                                 | •                            | •                         | •                                 |

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading and Bulk Unloading for Google BigQuery](#)”.

# SAS/ACCESS Interface to Greenplum: Supported Features

Here are the features that SAS/ACCESS Interface to Greenplum supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.7** Features by Host Environment for Greenplum

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5 <sup>1</sup>    |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>2</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |
| Solaris for SPARC              | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |

<sup>1</sup> In the current release, you can include SAS/ACCESS Interface to Greenplum with SAS Viya 3.5. However, there is not a data connector for Greenplum.

<sup>2</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for Greenplum](#)” on page 885.

# SAS/ACCESS Interface to Hadoop: Supported Features

Here are the features that SAS/ACCESS Interface to Hadoop supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.8** Features by Host Environment for Hadoop

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| 64-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |
| Solaris for SPARC              | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for Hadoop](#)” on page 911.

For detailed information about the following topics, see the SAS system requirements and configuration guide documents. They are available at <http://support.sas.com>:

- supported Hadoop versions, character data formats support, and required JAR files, environmental variables, and patches—along with other prerequisites, required setup, and considerations for SAS/ACCESS Interface to Hadoop

- how SAS/ACCESS Interface to Hadoop interacts with Hadoop through Hive

# SAS/ACCESS Interface to HAWQ: Supported Features

**Note:** Beginning in **SAS 9.4M8**, SAS/ACCESS Interface to HAWQ is no longer available. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

Here are the features that SAS/ACCESS Interface to HAWQ supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.9** Features by Host Environment for HAWQ

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5 <sup>1</sup>    |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                      | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium        | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux             | •                            | •                         | •                 | •                            | •                         | •                 |
| Solaris for SPARC        | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris           | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |

<sup>1</sup> In the current release, you can include SAS/ACCESS Interface to HAWQ with SAS Viya 3.5. However, there is not a data connector for HAWQ.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for HAWQ](#)” on page 940.

---

# SAS/ACCESS Interface to Impala: Supported Features

Here are the features that SAS/ACCESS Interface to Impala supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.10** Features by Host Environment for Impala

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                      | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux             | •                            | •                         | •                 | •                            | •                         | •                 |
| 64-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for Impala](#)”.

---

# SAS/ACCESS Interface to Informix: Supported Features

Here are the features that SAS/ACCESS Interface to Informix supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.11** Features by Host Environment for Informix

| Platform                       | SAS 9.4                      |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         |                   |
| 64-bit Linux                   | •                            | •                         |                   |
| Solaris for SPARC              | •                            | •                         |                   |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4.

## SAS/ACCESS Interface to JDBC: Supported Features

Here are the features that SAS/ACCESS Interface to JDBC supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.12** Features by Host Environment for JDBC

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| Solaris for SPARC        | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |

1 The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for JDBC](#)”.

## SAS/ACCESS Interface to Microsoft SQL Server: Supported Features

Here are the features that SAS/ACCESS Interface to Microsoft SQL Server supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.13** Features by Host Environment for Microsoft SQL Server

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| Solaris for SPARC        | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |

1 The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

If you have SAS 9.4, then SAS/ACCESS Interface to Microsoft SQL Server supports the DBLOAD procedure for all platforms except 32-bit and 64-bit Microsoft Windows. The DBLOAD procedure is not included with SAS Viya 3.5.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading with Microsoft SQL Server](#)”.

## SAS/ACCESS Interface to MySQL: Supported Features

Here are the features that SAS/ACCESS Interface to MySQL supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.14** Features by Host Environment for MySQL

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX <sup>1</sup>               | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>2</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| Solaris for SPARC        | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris           | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |

1 SAS/ACCESS Interface to MySQL on AIX is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

2 The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for MySQL](#)”.

## SAS/ACCESS Interface to Netezza: Supported Features

Here are the features that SAS/ACCESS Interface to Netezza supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

*Table 11.15 Features by Host Environment for Netezza*

| Platform                       | SAS 9.4                      |                           |                                   | SAS Viya 3.5 <sup>1</sup>    |                           |                                   |
|--------------------------------|------------------------------|---------------------------|-----------------------------------|------------------------------|---------------------------|-----------------------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support |
| AIX                            | •                            | •                         | •                                 |                              |                           |                                   |
| HP-UX for Itanium <sup>2</sup> | •                            | •                         | •                                 |                              |                           |                                   |

| Platform                 | SAS 9.4                      |                           |                                   | SAS Viya 3.5 <sup>1</sup>    |                           |                                   |
|--------------------------|------------------------------|---------------------------|-----------------------------------|------------------------------|---------------------------|-----------------------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load and Bulk-Unload Support |
| 64-bit Linux             | •                            | •                         | •                                 | •                            | •                         | •                                 |
| Solaris for SPARC        | •                            | •                         | •                                 |                              |                           |                                   |
| 64-bit Solaris           | •                            | •                         | •                                 |                              |                           |                                   |
| 32-bit Microsoft Windows | •                            | •                         | •                                 |                              |                           |                                   |
| 64-bit Microsoft Windows | •                            | •                         | •                                 |                              |                           |                                   |

- 1 In the current release, you can order SAS/ACCESS Interface to Netezza with SAS Viya 3.5. However, there is not a data connector for Netezza.
- 2 The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading and Unloading for Netezza](#)” on page 1061.

## SAS/ACCESS Interface to ODBC: Supported Features

Here are the features that SAS/ACCESS Interface to ODBC supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.16** Features by Host Environment for ODBC

| Platform | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|----------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX      | •                            | •                         |                   |                              |                           |                   |

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         |                   |                              |                           |                   |
| Linux on Power                 | •                            | •                         |                   | •                            | •                         |                   |
| 64-bit Linux                   | •                            | •                         |                   | •                            | •                         |                   |
| Solaris for SPARC              | •                            | •                         |                   |                              |                           |                   |
| 64-bit Solaris                 | •                            | •                         |                   |                              |                           |                   |
| 32-bit Microsoft Windows       | •                            | •                         | • <sup>2</sup>    |                              |                           |                   |
| 64-bit Microsoft Windows       | •                            | •                         | • <sup>2</sup>    | •                            | •                         | •                 |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

<sup>2</sup> Bulk-load support is available only with the Microsoft SQL Server driver on Microsoft Windows platforms.

If you have SAS 9.4, then SAS/ACCESS Interface to ODBC supports the DBLOAD procedure for all platforms. The DBLOAD procedure is not included with SAS Viya 3.5.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for ODBC](#)” on page 1096.

---

## SAS/ACCESS Interface to OLE DB: Supported Features

Here are the features that SAS/ACCESS Interface to OLE DB supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.17** Features by Host Environment for OLE DB

| Platform                 | SAS 9.4                      |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| 32-bit Microsoft Windows | •                            | •                         | •                 |
| 64-bit Microsoft Windows | •                            | •                         | •                 |

For information about these features, see “Methods for Accessing Relational Database Data” on page 4 and “Bulk Loading for OLE DB” on page 1121.

## SAS/ACCESS Interface to Oracle: Supported Features

Here are the features that SAS/ACCESS Interface to Oracle supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.18** Features by Host Environment for Oracle

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| Solaris for SPARC              | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| 64-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| z/OS <sup>2</sup>        | •                            | •                         | •                 |                              |                           |                   |

1 The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

2 SAS/ACCESS Interface to Oracle on z/OS is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

If you have SAS 9.4, then SAS/ACCESS Interface to Oracle supports the ACCESS and DBLOAD procedures for all platforms. These procedures are not included with SAS Viya 3.5.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for Oracle](#)” on page 1154.

## SAS/ACCESS Interface to PostgreSQL: Supported Features

Here are the features that SAS/ACCESS Interface to PostgreSQL supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.19** Features by Host Environment for PostgreSQL

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| Linux on Power           | •                            | •                         | •                 | •                            | •                         | •                 |
| 64-bit Linux             | •                            | •                         | •                 | •                            | •                         | •                 |
| Solaris for SPARC        | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris           | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows | •                            | •                         | •                 | •                            | •                         | •                 |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “Methods for Accessing Relational Database Data” on page 4 and “Bulk Loading and Unloading for PostgreSQL” on page 1186.

## SAS/ACCESS Interface to SAP ASE: Supported Features

Here are the features that SAS/ACCESS Interface to SAP ASE supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.20** Features by Host Environment for SAP ASE

| Platform | SAS 9.4                      |                           |                   | SAS Viya 3.5 <sup>1</sup>    |                           |                   |
|----------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX      | •                            | •                         | •                 |                              |                           |                   |

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5 <sup>1</sup>    |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| HP-UX for Itanium <sup>2</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| Solaris for SPARC              | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |

<sup>1</sup> In the current release, you can include SAS/ACCESS Interface to SAP ASE with SAS Viya 3.5. However, there is not a data connector for SAP ASE.

<sup>2</sup> The HP-UX platform is no longer available starting in [SAS 9.4M8](#). Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

If you have SAS 9.4, then SAS/ACCESS Interface to SAP ASE supports the ACCESS and DBLOAD procedures on all platforms. These procedures are not included with SAS Viya 3.5.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for SAP ASE](#)” on page 1211.

---

# SAS/ACCESS Interface to SAP HANA: Supported Features

Here are the features that SAS/ACCESS Interface to SAP HANA supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.21** Features by Host Environment for SAP HANA

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         | •                 | •                            | •                         | •                 |
| Solaris for SPARC              | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris                 | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows       | •                            | •                         | •                 |                              |                           |                   |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “Methods for Accessing Relational Database Data” on page 4 and “Bulk Loading for SAP HANA” on page 1240.

## SAS/ACCESS Interface to SAP IQ: Supported Features

Here are the features that SAS/ACCESS Interface to SAP IQ supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.22** Features by Host Environment for SAP IQ

| Platform                       | SAS 9.4                      |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |

| <b>SAS 9.4</b>           |                                     |                                  |                          |
|--------------------------|-------------------------------------|----------------------------------|--------------------------|
| <b>Platform</b>          | <b>SAS/ACCESS LIBNAME Statement</b> | <b>SQL Pass-Through Facility</b> | <b>Bulk-Load Support</b> |
| 64-bit Linux             | •                                   | •                                | •                        |
| Solaris for SPARC        | •                                   | •                                | •                        |
| 64-bit Solaris           | •                                   | •                                | •                        |
| 32-bit Microsoft Windows | •                                   | •                                | •                        |
| 64-bit Microsoft Windows | •                                   | •                                | •                        |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for SAP IQ](#)” on page 1263.

## SAS/ACCESS Interface to Snowflake: Supported Features

Here are the features that SAS/ACCESS Interface to Snowflake supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.23** Features by Host Environment for Snowflake

| <b>Platform</b>          | <b>SAS 9.4</b>                      |                                  |                                          | <b>SAS Viya 3.5</b>                 |                                  |                                          |
|--------------------------|-------------------------------------|----------------------------------|------------------------------------------|-------------------------------------|----------------------------------|------------------------------------------|
|                          | <b>SAS/ACCESS LIBNAME Statement</b> | <b>SQL Pass-Through Facility</b> | <b>Bulk-Load and Bulk-Unload Support</b> | <b>SAS/ACCESS LIBNAME Statement</b> | <b>SQL Pass-Through Facility</b> | <b>Bulk-Load and Bulk-Unload Support</b> |
| 64-bit Linux             | •                                   | •                                | •                                        | •                                   | •                                | •                                        |
| 64-bit Microsoft Windows | •                                   | •                                | •                                        | •                                   | •                                | •                                        |

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading and Unloading for Snowflake](#)”.

---

## SAS/ACCESS Interface to Spark: Supported Features

Here are the features that SAS/ACCESS Interface to Spark supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

*Table 11.24 Features by Host Environment for Spark*

| Platform     | SAS 9.4                      |                           |                   |
|--------------|------------------------------|---------------------------|-------------------|
|              | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| 64-bit Linux | •                            | •                         | •                 |

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Bulk Loading for Spark](#)”.

---

## SAS/ACCESS Interface to Teradata: Supported Features

Here are the features that SAS/ACCESS Interface to Teradata supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

*Table 11.25 Features by Host Environment for Teradata*

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         | •                 |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         | •                 |                              |                           |                   |

| Platform                 | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                          | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| 64-bit Linux             | •                            | •                         | •                 | •                            | •                         | •                 |
| Solaris for SPARC        | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Solaris           | •                            | •                         | •                 |                              |                           |                   |
| 32-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| 64-bit Microsoft Windows | •                            | •                         | •                 |                              |                           |                   |
| z/OS <sup>2</sup>        | •                            | •                         | •                 |                              |                           |                   |

1 The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

2 SAS/ACCESS Interface to Teradata on z/OS is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4 and “[Maximizing Teradata Load and Read Performance](#)” on page 1337.

---

## SAS/ACCESS Interface to Vertica: Supported Features

Here are the features that SAS/ACCESS Interface to Vertica supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.26** Features by Host for SAS/ACCESS Interface to Vertica

| Platform                       | SAS 9.4                      |                           |                   | SAS Viya 3.5                 |                           |                   |
|--------------------------------|------------------------------|---------------------------|-------------------|------------------------------|---------------------------|-------------------|
|                                | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
| AIX                            | •                            | •                         |                   |                              |                           |                   |
| HP-UX for Itanium <sup>1</sup> | •                            | •                         |                   |                              |                           |                   |
| 64-bit Linux                   | •                            | •                         |                   | •                            | •                         |                   |
| Solaris for SPARC              | •                            | •                         |                   |                              |                           |                   |
| 64-bit Microsoft Windows       | •                            | •                         |                   |                              |                           |                   |

<sup>1</sup> The HP-UX platform is no longer available starting in SAS 9.4M8. Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

For information about these features, see “[Methods for Accessing Relational Database Data](#)” on page 4.

## SAS/ACCESS Interface to Yellowbrick: Supported Features

Here are the features that SAS/ACCESS Interface to Yellowbrick supports. To find out which versions of your DBMS are supported, see your system requirements documentation.

**Table 11.27** Features by Host Environment for Yellowbrick

| Platform                 | SAS/ACCESS LIBNAME Statement | SQL Pass-Through Facility | Bulk-Load Support |
|--------------------------|------------------------------|---------------------------|-------------------|
| 64-bit AIX               | •                            | •                         | •                 |
| 64-bit Linux             | •                            | •                         | •                 |
| Solaris for SPARC        | •                            | •                         | •                 |
| 64-bit Microsoft Windows | •                            | •                         | •                 |

For information about these features, see Chapter 41, “SAS/ACCESS Interface to Yellowbrick,” on page [1399](#).



# The LIBNAME Statement

---

|                                                                     |            |
|---------------------------------------------------------------------|------------|
| <b>Overview: LIBNAME Statement for Relational Databases .....</b>   | <b>127</b> |
| Assigning Librefs .....                                             | 127        |
| Sorting Data .....                                                  | 127        |
| Using SAS Functions .....                                           | 127        |
| <b>Assigning a Libref Interactively .....</b>                       | <b>128</b> |
| <b>Dictionary .....</b>                                             | <b>129</b> |
| LIBNAME Statement: External Databases .....                         | 129        |
| ACCESS= LIBNAME Statement Option .....                              | 134        |
| ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS= LIBNAME Statement Option ..... | 135        |
| ADJUST_NCHAR_COLUMN_LENGTHS= LIBNAME Statement Option .....         | 136        |
| ALLOW_LARGE_RESULTS= LIBNAME Statement Option .....                 | 138        |
| ALLOWED_SQLCODES= LIBNAME Statement Option .....                    | 139        |
| ANALYZE= LIBNAME Statement Option .....                             | 139        |
| AUTHDOMAIN= LIBNAME Statement Option .....                          | 140        |
| AUTHID= LIBNAME Statement Option .....                              | 142        |
| AUTOCOMMIT= LIBNAME Statement Option .....                          | 142        |
| BL_ACCOUNTNAME= LIBNAME Statement Option .....                      | 143        |
| BL_APPLICATIONID= LIBNAME Statement Option .....                    | 144        |
| BL_AWS_CONFIG_FILE= LIBNAME Statement Option .....                  | 145        |
| BL_AWS_PROFILE_NAME= LIBNAME Statement Option .....                 | 146        |
| BL_BUCKET= LIBNAME Statement Option .....                           | 147        |
| BL_COMPRESS= LIBNAME Statement Option .....                         | 147        |
| BL_CONFIG= LIBNAME Statement Option .....                           | 148        |
| BL_DEFAULT_DIR= LIBNAME Statement Option .....                      | 149        |
| BL_DELETE_DATAFILE= LIBNAME Statement Option .....                  | 151        |
| BL_DELIMITER= LIBNAME Statement Option .....                        | 152        |
| BL_DNSSUFFIX= LIBNAME Statement Option .....                        | 152        |
| BL_ENCKEY= LIBNAME Statement Option .....                           | 153        |
| BL_FILESYSTEM= LIBNAME Statement Option .....                       | 154        |
| BL_FOLDER= LIBNAME Statement Option .....                           | 155        |
| BL_FORMAT= LIBNAME Statement Option .....                           | 156        |
| BL_HOST= LIBNAME Statement Option .....                             | 157        |
| BL_IDENTITY= LIBNAME Statement Option .....                         | 158        |
| BL_KEEPIDENTITY= LIBNAME Statement Option .....                     | 159        |
| BL_KEEPNULLS= LIBNAME Statement Option .....                        | 159        |

|                                                          |     |
|----------------------------------------------------------|-----|
| BL_KEY= LIBNAME Statement Option .....                   | 160 |
| BL_LOG= LIBNAME Statement Option .....                   | 161 |
| BL_MAXERRORS= LIBNAME Statement Option .....             | 161 |
| BL_NUM_READ_THREADS= LIBNAME Statement Option .....      | 162 |
| BL_OPTIONS= LIBNAME Statement Option .....               | 163 |
| BL_PORT= LIBNAME Statement Option .....                  | 164 |
| BL_RECOVERABLE= LIBNAME Statement Option .....           | 165 |
| BL_REGION= LIBNAME Statement Option .....                | 166 |
| BL_SECRET= LIBNAME Statement Option .....                | 166 |
| BL_TIMEOUT= LIBNAME Statement Option .....               | 167 |
| BL_TOKEN= LIBNAME Statement Option .....                 | 168 |
| BL_USE_ESCAPE= LIBNAME Statement Option .....            | 169 |
| BL_USE_LOG= LIBNAME Statement Option .....               | 170 |
| BL_USE_SSL= LIBNAME Statement Option .....               | 171 |
| BULKLOAD= LIBNAME Statement Option .....                 | 171 |
| BULKUNLOAD= LIBNAME Statement Option .....               | 173 |
| CAST= LIBNAME Statement Option .....                     | 174 |
| CAST_OVERHEAD_MAXPERCENT= LIBNAME Statement Option ..... | 176 |
| CELLPROP= LIBNAME Statement Option .....                 | 177 |
| CHAR_AS_NCHAR= LIBNAME Statement Option .....            | 178 |
| CLIENT_ENCODING= LIBNAME Statement Option .....          | 179 |
| CLIENT_ID= LIBNAME Statement Option .....                | 180 |
| CLIENT_SECRET= LIBNAME Statement Option .....            | 180 |
| COMMAND_TIMEOUT= LIBNAME Statement Option .....          | 181 |
| CONNECTION= LIBNAME Statement Option .....               | 182 |
| CONNECTION_GROUP= LIBNAME Statement Option .....         | 188 |
| CONOPTS= LIBNAME Statement Option .....                  | 190 |
| CURSOR_TYPE= LIBNAME Statement Option .....              | 191 |
| DATETIME2= LIBNAME Statement Option .....                | 193 |
| DB_LENGTH_SEMANTICS_BYTE= LIBNAME Statement Option ..... | 194 |
| DB_OBJECTS= LIBNAME Statement Option .....               | 195 |
| DBCLIENT_ENCODING_FIXED= LIBNAME Statement Option .....  | 196 |
| DBCLIENT_MAX_BYTES= LIBNAME Statement Option .....       | 197 |
| DBCOMMIT= LIBNAME Statement Option .....                 | 199 |
| DBCONINIT= LIBNAME Statement Option .....                | 200 |
| DBCONTERM= LIBNAME Statement Option .....                | 202 |
| DBCREATE_TABLE_EXTERNAL= LIBNAME Statement Option .....  | 204 |
| DBCREATE_TABLE_LOCATION= LIBNAME Statement Option .....  | 205 |
| DBCREATE_TABLE_OPTS= LIBNAME Statement Option .....      | 207 |
| DBGEN_NAME= LIBNAME Statement Option .....               | 208 |
| DBINDEX= LIBNAME Statement Option .....                  | 210 |
| DBLIBINIT= LIBNAME Statement Option .....                | 211 |
| DBLIBTERM= LIBNAME Statement Option .....                | 212 |
| DBLINK= LIBNAME Statement Option .....                   | 214 |
| DBMAX_TEXT= LIBNAME Statement Option .....               | 214 |
| DBMSTEMP= LIBNAME Statement Option .....                 | 215 |
| DBNULLKEYS= LIBNAME Statement Option .....               | 218 |
| DBNULLWHERE= LIBNAME Statement Option .....              | 219 |
| DBPROMPT= LIBNAME Statement Option .....                 | 220 |
| DBSASLABEL= LIBNAME Statement Option .....               | 222 |
| DBSERVER_ENCODING_FIXED= LIBNAME Statement Option .....  | 223 |
| DBSERVER_MAX_BYTES= LIBNAME Statement Option .....       | 225 |
| DBSLICEPARM= LIBNAME Statement Option .....              | 226 |
| DEFAULT_AUTH_PLUGIN= LIBNAME Statement Option .....      | 229 |

|                                                               |     |
|---------------------------------------------------------------|-----|
| DEFER= LIBNAME Statement Option .....                         | 230 |
| DEGREE= LIBNAME Statement Option .....                        | 231 |
| DELETE_MULT_ROWS= LIBNAME Statement Option .....              | 232 |
| DIMENSION= LIBNAME Statement Option .....                     | 233 |
| DIRECT_EXE= LIBNAME Statement Option .....                    | 233 |
| DIRECT_SQL= LIBNAME Statement Option .....                    | 234 |
| DRIVER_TRACE= LIBNAME Statement Option .....                  | 237 |
| DRIVER_TRACEFILE= LIBNAME Statement Option .....              | 238 |
| DRIVER_TRACEOPTIONS= LIBNAME Statement Option .....           | 238 |
| DRIVER_VENDOR= LIBNAME Statement Option .....                 | 239 |
| ENABLE_BULK= LIBNAME Statement Option .....                   | 240 |
| ERRLIMIT= LIBNAME Statement Option .....                      | 241 |
| ESCAPE_BACKSLASH= LIBNAME Statement Option .....              | 242 |
| FASTEXPORT= LIBNAME Statement Option .....                    | 242 |
| FASTLOAD= LIBNAME Statement Option .....                      | 244 |
| FETCH_IDENTITY= LIBNAME Statement Option .....                | 246 |
| HDFS_PRINCIPAL= LIBNAME Option LIBNAME Statement Option ..... | 247 |
| IGNORE_BASELINE= LIBNAME Statement Option .....               | 247 |
| IGNORE_READ_ONLY_COLUMNS= LIBNAME Statement Option .....      | 249 |
| IMPALA_PRINCIPAL= LIBNAME Statement Option .....              | 250 |
| IN= LIBNAME Statement Option .....                            | 251 |
| INSERT_SQL= LIBNAME Statement Option .....                    | 252 |
| INSERTBUFF= LIBNAME Statement Option .....                    | 253 |
| INTERFACE= LIBNAME Statement Option .....                     | 255 |
| KEYSET_SIZE= LIBNAME Statement Option .....                   | 255 |
| LARGE_RESULTS_DATASET= LIBNAME Statement Option .....         | 256 |
| LARGE_RESULTS_EXPIRATION_TIME= LIBNAME Statement Option ..... | 257 |
| LOCATION= LIBNAME Statement Option .....                      | 258 |
| LOCKTABLE= LIBNAME Statement Option .....                     | 259 |
| LOCKTIME= LIBNAME Statement Option .....                      | 259 |
| LOCKWAIT= LIBNAME Statement Option .....                      | 260 |
| LOGDB= LIBNAME Statement Option .....                         | 260 |
| LOGIN_TIMEOUT= LIBNAME Statement Option .....                 | 262 |
| MAX_BINARY_LEN= LIBNAME Statement Option .....                | 263 |
| MAX_CHAR_LEN= LIBNAME Statement Option .....                  | 263 |
| MAX_CONNECTS= LIBNAME Statement Option .....                  | 264 |
| MAX_STRING_SIZE= LIBNAME Statement Option .....               | 264 |
| MODE= LIBNAME Statement Option .....                          | 265 |
| MULTI_DATASRC_OPT= LIBNAME Statement Option .....             | 267 |
| MULTISTMT= LIBNAME Statement Option .....                     | 269 |
| OR_BINARY_DOUBLE= LIBNAME Statement Option .....              | 270 |
| OR_ENABLE_INTERRUPT= LIBNAME Statement Option .....           | 271 |
| OR_UPD_NOWHERE= LIBNAME Statement Option .....                | 271 |
| OVERRIDE_RESP_LEN= LIBNAME Statement Option .....             | 272 |
| PACKETSIZE= LIBNAME Statement Option .....                    | 274 |
| PARMDEFAULT= LIBNAME Statement Option .....                   | 274 |
| PARMSTRING= LIBNAME Statement Option .....                    | 275 |
| PARTITION_KEY= LIBNAME Statement Option .....                 | 277 |
| POST_DML_STMT_OPTS= LIBNAME Statement Option .....            | 278 |
| POST_STMT_OPTS= LIBNAME Statement Option .....                | 279 |
| PRESERVE_COL_NAMES= LIBNAME Statement Option .....            | 280 |
| PRESERVE_COMMENTS= LIBNAME Statement Option .....             | 282 |
| PRESERVE_GUID= LIBNAME Statement Option .....                 | 284 |
| PRESERVE_TAB_NAMES= LIBNAME Statement Option .....            | 289 |

|                                                           |     |
|-----------------------------------------------------------|-----|
| PRESERVE_USER= LIBNAME Statement Option .....             | 292 |
| PROGRAM_NAME= LIBNAME Statement Option .....              | 293 |
| PROPERTIES= LIBNAME Statement Option .....                | 294 |
| PROXY= LIBNAME Statement Option .....                     | 296 |
| QUALIFIER= LIBNAME Statement Option .....                 | 296 |
| QUALIFY_ROWS= LIBNAME Statement Option .....              | 298 |
| QUERY_BAND= LIBNAME Statement Option .....                | 299 |
| QUERY_TIMEOUT= LIBNAME Statement Option .....             | 300 |
| QUOTE_CHAR= LIBNAME Statement Option .....                | 300 |
| QUOTED_IDENTIFIER= LIBNAME Statement Option .....         | 302 |
| READ_ISOLATION_LEVEL= LIBNAME Statement Option .....      | 302 |
| READ_LOCK_TYPE= LIBNAME Statement Option .....            | 303 |
| READ_METHOD= LIBNAME Statement Option .....               | 305 |
| READ_MODE= LIBNAME Statement Option .....                 | 306 |
| READ_MODE_WAIT= LIBNAME Statement Option .....            | 307 |
| READBUFF= LIBNAME Statement Option .....                  | 308 |
| REFRESH_TOKEN= LIBNAME Statement Option .....             | 310 |
| REMOTE_DBTYPE= LIBNAME Statement Option .....             | 311 |
| REREAD_EXPOSURE= LIBNAME Statement Option .....           | 312 |
| RESULTS= LIBNAME Statement Option .....                   | 313 |
| SAS_DBMS_AUTOMETADATA= LIBNAME Statement Option .....     | 314 |
| SCANSTRINGCOLUMNS= LIBNAME Statement Option .....         | 315 |
| SCHEMA= LIBNAME Statement Option .....                    | 316 |
| SCRATCH_DB= LIBNAME Option LIBNAME Statement Option ..... | 319 |
| SESSIONS= LIBNAME Statement Option .....                  | 320 |
| SHOW_SYNONYMS= LIBNAME Statement Option .....             | 321 |
| SLEEP= LIBNAME Statement Option .....                     | 322 |
| SPOOL= LIBNAME Statement Option .....                     | 323 |
| SQL_FUNCTIONS= LIBNAME Statement Option .....             | 324 |
| SQL_FUNCTIONS_COPY= LIBNAME Statement Option .....        | 328 |
| SQL_OJ_ANSI= LIBNAME Statement Option .....               | 329 |
| SQLGENERATION= LIBNAME Statement Option .....             | 330 |
| SSL_CA= LIBNAME Statement Option .....                    | 331 |
| SSL_CERT= LIBNAME Statement Option .....                  | 332 |
| SSL_CIPHER= LIBNAME Statement Option .....                | 333 |
| SSL_KEY= LIBNAME Statement Option .....                   | 333 |
| SSLMODE= LIBNAME Statement Option .....                   | 334 |
| STRINGDATES= LIBNAME Statement Option .....               | 336 |
| SUB_CHAR= LIBNAME Statement Option .....                  | 337 |
| SYNONYMS= LIBNAME Statement Option .....                  | 338 |
| TABLE_TYPE= LIBNAME Statement Option .....                | 339 |
| TEMP_CTAS= LIBNAME Statement Option .....                 | 340 |
| TEMPORAL_QUALIFIER= LIBNAME Statement Option .....        | 341 |
| TENACITY= LIBNAME Statement Option .....                  | 343 |
| TPT= LIBNAME Statement Option .....                       | 344 |
| TPT_DATA_ENCRYPTION= LIBNAME Statement Option .....       | 345 |
| TPT_MAX_SESSIONS= LIBNAME Statement Option .....          | 346 |
| TPT_MIN_SESSIONS= LIBNAME Statement Option .....          | 347 |
| TPT_UNICODE_PASSTHRU= LIBNAME Statement Option .....      | 348 |
| TR_ENABLE_INTERRUPT= LIBNAME Statement Option .....       | 349 |
| TRACE= LIBNAME Statement Option .....                     | 350 |
| TRACEFILE= LIBNAME Statement Option .....                 | 351 |
| TRACEFLAGS= LIBNAME Statement Option .....                | 352 |
| TRANSCODE_FAIL= LIBNAME Statement Option .....            | 353 |

|                                                                 |     |
|-----------------------------------------------------------------|-----|
| UPDATE_ISOLATION_LEVEL= LIBNAME Statement Option .....          | 354 |
| UPDATE_LOCK_TYPE= LIBNAME Statement Option .....                | 355 |
| UPDATE_MODE_WAIT= LIBNAME Option LIBNAME Statement Option ..... | 356 |
| UPDATE_MULT_ROWS= LIBNAME Statement Option .....                | 357 |
| UPDATE_SQL= LIBNAME Statement Option .....                      | 358 |
| UPDATEBUFF= LIBNAME Statement Option .....                      | 359 |
| USE_DATADIRECT= LIBNAME Statement Option .....                  | 360 |
| USE_INFORMATION_SCHEMA= LIBNAME Statement Option .....          | 360 |
| USE_ODBC_CL= LIBNAME Statement Option .....                     | 361 |
| UTILCONN_TRANSIENT= LIBNAME Statement Option .....              | 362 |
| WARN_BIGINT= LIBNAME Statement Option .....                     | 363 |

# Overview: LIBNAME Statement for Relational Databases

## Assigning Librefs

The SAS/ACCESS LIBNAME statement extends the SAS global LIBNAME statement so that you can assign a libref to a relational DBMS. This feature lets you reference a DBMS object directly in a DATA step or SAS procedure. You can use it to read from and write to a DBMS object as if it were a SAS data set. You can associate a SAS libref with a relational DBMS database, schema, server, or group of tables and views.

For details about the syntax, see “[LIBNAME Statement: External Databases](#)” on [page 129](#). For the engine name, connection options, and LIBNAME options for your SAS/ACCESS interface, see the DBMS-specific reference section for your SAS/ACCESS interface.

For details about specifying and naming librefs, see “[Rules for Most SAS Names](#)” in [SAS Programmer’s Guide: Essentials](#).

## Sorting Data

When you use the SAS/ACCESS LIBNAME statement to associate a libref with relational DBMS data, you might observe some behavior that differs from that of normal SAS librefs. Because these librefs refer to database objects, such as tables and views, they are stored in the format of your DBMS. DBMS format differs from the format of normal SAS data sets. This is helpful to remember when you access and work with DBMS data.

For example, you can sort the observations in a normal SAS data set and store the output in another data set. However, in a relational DBMS, sorting data often has no effect on how it is stored. Because you cannot depend on your data being sorted in the DBMS, you must sort the data at the time of query. Also, when you sort DBMS

data, results might vary depending on whether your DBMS places data with NULL values (which SAS translates into missing values) at the beginning or the end of the result set.

---

## Using SAS Functions

When you use librefs that refer to DBMS data with SAS functions, some functions might return a value that differs from what is returned when you use the functions with normal SAS data sets. For example, the PATHNAME function might return a blank value. For a normal SAS libref, a blank value means that the libref is not valid. However, for a libref associated with a DBMS object, a blank value means only that there is no path name associated with the libref.

Usage of some functions might also vary. For example, the LIBNAME function can accept an optional *SAS-library* argument. When you use the LIBNAME function to assign or unassign a libref that refers to DBMS data, you omit this argument. For full details about how to use SAS functions, see the [SAS Functions and CALL Routines: Reference](#).

---

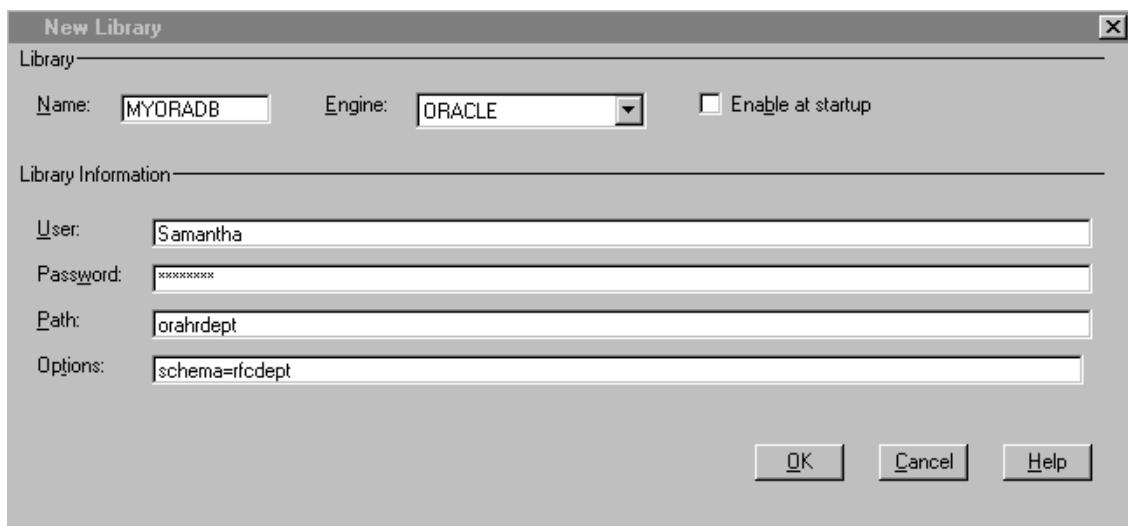
## Assigning a Libref Interactively

An easy way to associate a libref with a relational DBMS is to use the New Library window in the SAS windowing environment. One method to open this window is to issue the DMLIBASSIGN command from your SAS session's command box or command line. You can also open the window by clicking the file cabinet icon in the **SAS Explorer** toolbar. Below, the user Samantha assigns a libref MYORADB to an Oracle database that the SQL\*Net alias ORAHRDEPT references. By using the SCHEMA= LIBNAME option, Samantha can access database objects that another user owns.

---

**Note:** SAS windowing environment is not applicable to SAS Viya.

---

**Figure 12.1** New Library Window

Here is how to use the features of the New Library window.

- **Name:** Enter the libref that you want to assign to a SAS library or a relational DBMS.
- **Engine:** Click the down arrow to select a name from the pull-down listing.
- **Enable at startup:** Click this if you want the specified libref to be assigned automatically when you open a SAS session.
- **Library Information:** These fields represent the SAS/ACCESS connection options and vary according to the SAS/ACCESS engine that you specify. Enter the appropriate information for your site's DBMS. The **Options** field lets you enter SAS/ACCESS LIBNAME options. Use blanks to separate multiple options.
- **OK:** Click this button to assign the libref, or click **Cancel** to exit the window without assigning a libref.

---

## Dictionary

---

### LIBNAME Statement: External Databases

Associates a SAS libref with a DBMS database, schema, server, or a group of tables and views.

Valid in: anywhere

See: ["Overview: LIBNAME Statement for Relational Databases"](#)

## Syntax

- Form 1: **LIBNAME** *libref engine-name*  
           *<SAS/ACCESS-connection-options>*  
           *<SAS/ACCESS-LIBNAME-options>;*
- Form 2: **LIBNAME** *libref CLEAR | \_ALL\_ CLEAR;*
- Form 3: **LIBNAME** *libref LIST | \_ALL\_ LIST;*

## Required Arguments

### ***libref***

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views. Like the global SAS LIBNAME statement, the SAS/ACCESS LIBNAME statement creates shortcuts or nicknames for data storage locations. A SAS libref is an alias for a virtual or physical directory. A SAS/ACCESS libref is an alias for the DBMS database, schema, or server where your tables and views are stored.

### ***engine-name***

specifies the SAS/ACCESS engine name for your DBMS, such as hadoop. The engine name is required. Because the SAS/ACCESS LIBNAME statement associates a libref with a SAS/ACCESS engine that supports connections to a particular DBMS, it requires a DBMS-specific engine name. See the DBMS-specific reference section for details.

### **CLEAR**

disassociates one or more currently assigned librefs.

Specify *libref* to disassociate a single libref. Specify *\_ALL\_* to disassociate all currently assigned librefs.

### **\_ALL\_**

specifies that the CLEAR or LIST argument applies to all currently assigned librefs.

### **LIST**

writes the attributes of one or more SAS/ACCESS libraries or SAS libraries to the SAS log.

Specify *libref* to list the attributes of a single SAS/ACCESS library or SAS library. Specify *\_ALL\_* to list the attributes of all libraries that have librefs in your current session.

## Optional Arguments

### **SAS/ACCESS-connection-options**

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. These arguments are different for each database. For example, to connect to a Hadoop database, your connection options are USER=, PASSWORD=, and SERVER=:

```
libname mydblib hadoop user=myusr1 password=mypwd1 server=hadoopsrv;
```

If the connection options contain characters that are not allowed in SAS names, enclose the values of the arguments in quotation marks. On some DBMSs, if you specify the appropriate system options or environment variables for your

database, you can omit the connection options. For connection option details, see the DBMS-specific information for your SAS/ACCESS interface.

#### **SAS/ACCESS-LIBNAME-options**

define how DBMS objects are processed by SAS. Some LIBNAME options can enhance performance; others determine locking or naming behavior. For example, the PRESERVE\_COL\_NAMES= option lets you specify whether to preserve spaces, special characters, and mixed case in DBMS column names when creating tables. The availability and default behavior of many of these options are DBMS-specific. See the DBMS-specific reference section for LIBNAME options that are available for your SAS/ACCESS interface. For general information, see [LIBNAME Options for Relational Databases on page 134](#).

## Details

### Form 1: Using Data from a DBMS

You can use a LIBNAME statement to read from and write to a DBMS table or view as if it were a SAS data set.

For example, in MYDBLIB.EMPLOYEES\_Q2, MYDBLIB is a SAS libref that points to a particular group of DBMS objects, and EMPLOYEES\_Q2 is a DBMS table name. When you specify MYDBLIB.EMPLOYEES\_Q2 in a DATA step or procedure, you dynamically access the DBMS table. SAS supports reading, updating, creating, and deleting DBMS tables dynamically.

### Form 2: Disassociating a Libref from a SAS Library

To disassociate or clear a libref from a DBMS, use a LIBNAME statement. Specify the libref (for example, MYDBLIB) and the CLEAR option as shown here:

```
libname mydblib CLEAR;
```

You can clear a single specified libref or all current librefs.

The database engine disconnects from the database and closes any free threads or resources that are associated with that libref's connection.

### Form 3: Writing SAS Library Attributes to the SAS Log

Use a LIBNAME statement to write the attributes of one or more SAS/ACCESS libraries or SAS libraries to the SAS log. Specify *libref* to list the attributes of a single SAS/ACCESS library or SAS library, as shown below.

```
libname mydblib LIST;
```

Specify \_ALL\_ to list the attributes of all libraries that have librefs in your current session.

## SQL Views with Embedded LIBNAME Statements

With SAS software, you can embed LIBNAME statements in the definition of an SQL view. This means that you can store a LIBNAME statement in an SQL view that contains all information that is required to connect to a DBMS. Whenever the SQL view is read, PROC SQL uses the embedded LIBNAME statement to assign a libref. After the view has been processed, PROC SQL unassigns the libref.

In this example, an SQL view of the Hadoop table Dept is created. Whenever you use this view in a SAS program, the Hadlib library is assigned. The library uses the connection information (user name, password, and data source) that is provided in the embedded LIBNAME statement.

```
proc sql;
  create view sasuser.myview as
    select dname from hadlib.dept
    using libname hadlib hadoop
      user=scott pw=tiger datasrc=hadsrv;
quit;
```

---

**Note:** You can use the USING LIBNAME syntax to embed LIBNAME statements in SQL views. For more information about the USING LIBNAME syntax, see the [SAS SQL Procedure User's Guide](#).

---

## Assigning a Libref with a SAS/ACCESS LIBNAME Statement

This statement creates a libref, MYDBLIB, that uses the SAS/ACCESS Interface to Hadoop.

```
libname mydblib hadoop server=hadoopsrv user=user1 password=mypwd1;
```

The statement below associates the SAS libref MYDBLIB with a Hadoop database. You specify the SCHEMA= option in the SAS/ACCESS LIBNAME statement to connect to the Hadoop schema in which the data resides. In this example, Hadoop schemas reside in a database.

```
libname mydblib hadoop user=myusr1 password=mypwd1
  server=hadoopsrv schema=hrdept;
```

The Hadoop database contains a number of DBMS objects, including several tables, such as Staff. After you assign the libref, you can reference the table like a SAS data set and use it as a data source in any DATA step or SAS procedure. In the SQL procedure statement below, Mydblib.Staff is the two-level SAS name for the Staff table in the Hrdept schema.

```
proc sql;
  select idnum, lname
  from mydblib.staff
  where state='NY'
  order by lname;
quit;
```

You can use the DBMS data to create a SAS data set.

```
data newds;
  set mydblib.staff(keep=idnum lname fname);
run;
```

You can also use the libref and data set with any other SAS procedure. This statement prints the information in the Staff table.

```
proc print data=mydblib.staff;
run;
```

This statement lists the database objects in the Mydblib library.

```
proc datasets library=mydblib;
quit;
```

## Using the Prompting Window When Specifying LIBNAME Options

This statement uses the **DBPROMPT=** LIBNAME option to cause the DBMS connection prompting window to appear and prompt you for connection information.

---

**Note:** This information pertains only to the windowing environment and is not applicable to SAS Viya.

---

```
libname mydblib oracle dbprompt=yes;
```

When you use this option, you enter connection information into the fields in a prompting window rather than in the LIBNAME statement.

You can add the **DEFER= NO** LIBNAME option to make the prompting window appear when the libref is assigned rather than when the table is opened.

```
libname mydblib oracle dbprompt=yes defer=no;
```

## Assigning a Libref to a Remote DBMS

SAS/CONNECT (single-user) and SASSHARE (multiple user) software give you access to data by means of *remote library services* (RLS). RLS lets you access your data on a remote machine as if it were local. For example, it permits a graphical interface to reside on the local machine. The data remains on the remote machine.

This access is given to data stored in many types of SAS files. Examples include external databases (through the SAS/ACCESS LIBNAME statement and views that are created with it) and SAS data views (views that are created with PROC SQL, the DATA step, and SAS/ACCESS software). RLS lets you access SAS data sets, SAS views, and relational DBMS data that SAS/ACCESS LIBNAME statements specify. For more information, see the discussion about remote library services in the [SASSHARE User's Guide](#).

You can use RLS to update relational DBMS tables that are referenced with the SAS/ACCESS LIBNAME statement.

In the next example, the SAS/ACCESS LIBNAME statement makes a connection to a DB2 database that resides on the remote SASSHARE server REMOS390. This LIBNAME statement is submitted in a local SAS session. The SAS/ACCESS engine name is specified in the remote option RENGINE=. The DB2 connection option and any LIBNAME options are specified in the remote option ROPTIONS=. Options are separated by a blank space. RLSDB2.EMPLOYEES is a SAS data set that references the DB2 table EMPLOYEES.

```
libname rlsdb2 rengine=db2 server=remos390
```

```

      options="ssid=db2a authid=testid";
proc print data=rlsdb2.employees;
run;

```

## LIBNAME Options for Relational Databases

When you specify an option in the LIBNAME statement, it applies to all objects (such as tables and views) in the database that the libref represents. For information about options that you specify on individual SAS data sets, see [“About the Data Set Options for Relational Databases” on page 370](#). For general information, see [“LIBNAME Statement: External Databases” on page 129](#). See the DBMS-specific reference section for LIBNAME options that are available for your SAS/ACCESS interface.

Many LIBNAME options are also available for use with the SQL pass-through facility. See the section on the SQL pass-through facility in the documentation for your SAS/ACCESS interface to determine which LIBNAME options are available in the SQL pass-through facility for your DBMS. For general information, see [“SQL Pass-Through Facility” on page 690](#).

When a like-named option is specified in both the LIBNAME statement and after a data set name, SAS uses the value that is specified on the data set name.

## ACCESS= LIBNAME Statement Option

Determines the access level with which a libref connection is opened.

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                         |
| Category:    | Data Access                                                                                                                                                                                                                                                                                                                                          |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                  |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |

## Syntax

**ACCESS=READONLY**

## Syntax Description

### **READONLY**

specifies that you can read but not update tables and views.

---

## Details

Using this option prevents writing to the DBMS. If this option is omitted, you can read and update tables and views if you have the necessary DBMS privileges.

---

## ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS = LIBNAME Statement Option

Specifies whether to adjust the lengths of CHAR or VARCHAR data type columns with byte semantic column lengths.

|               |                                                                                                                                                                                                                                                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                  |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                              |
| Alias:        | EXPAND_BYTE_SEMANTIC_COLUMN_LENGTHS= [Oracle]                                                                                                                                                                                                                                                                                 |
| Defaults:     | based on the value of DBCLIENT_MAX_BYTES=<br>NO [Oracle. This can be changed dynamically]                                                                                                                                                                                                                                     |
| Interactions: | When the DBCLIENT_MAX_BYTES= value is greater than 1,<br>ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=YES. When<br>DBCLIENT_MAX_BYTES=1,ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=NO.<br>When ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=YES and<br>DBSERVER_ENCODING_FIXED=YES, this changes the character length calculation.<br>(See "Details".) |
| Supports:     | NLS                                                                                                                                                                                                                                                                                                                           |
| Data source:  | Oracle                                                                                                                                                                                                                                                                                                                        |
| See:          | <a href="#">SAS/ACCESS LIBNAME options for NLS</a>                                                                                                                                                                                                                                                                            |

---

## Syntax

**ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS=YES | NO**

## Syntax Description

### **YES**

indicates that column lengths are based on the number of characters multiplied by the DBCLIENT\_MAX\_BYTES= value. When the DBCLIENT\_MAX\_BYTES= value is greater than 1,  
ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS=YES.

**NO**

indicates that any column lengths that byte semantics specify on the server are used “as is” on the client. If DBCLIENT\_MAX\_BYTES=1, ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS=NO.

## Details

When ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS=YES and DBSERVER\_ENCODING\_FIXED=YES, character length is calculated as byte length divided by the DBSERVER\_MAX\_BYTES= value. When ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS=YES and DBSERVER\_ENCODING\_FIXED=NO, character lengths are not adjusted.

## Example: Adjust Client-Encoded Column Lengths

When ADJUST\_BYTE\_SEMANTIC\_COLUMN\_LENGTHS=YES, column lengths that byte semantics creates are adjusted with client encoding, as shown in this example.

```
libname x3 &engine &connopt ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=YES;
proc contents data=x3.char_sem; run;
proc contents data=x3.nchar_sem; run;
proc contents data=x3.byte_sem; run;
proc contents data=x3.mixed_sem; run;
```

In this example, various options have different values.

```
libname x5 &engine &connopt ADJUST_NCHAR_COLUMN_LENGTHS=NO
ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=NO DBCLIENT_MAX_BYTES=3;
proc contents data=x5.char_sem; run;
proc contents data=x5.nchar_sem; run;
proc contents data=x5.byte_sem; run;
proc contents data=x5.mixed_sem; run;
```

This example also uses different values for the various options.

```
libname x6 &engine &connopt ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=YES
ADJUST_NCHAR_COLUMN_LENGTHS=YES DBCLIENT_MAX_BYTES=3;
proc contents data=x6.char_sem; run;
proc contents data=x6.nchar_sem; run;
proc contents data=x6.byte_sem; run;
proc contents data=x6.mixed_sem; run;
```

## ADJUST\_NCHAR\_COLUMN\_LENGTHS= LIBNAME Statement Option

Specifies whether to adjust the lengths of NCHAR or NVARCHAR data type columns.

|              |                                                    |
|--------------|----------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                       |
| Category:    | Data Set Control                                   |
| Default:     | YES                                                |
| Supports:    | NLS                                                |
| Data source: | Oracle                                             |
| See:         | <a href="#">SAS/ACCESS LIBNAME options for NLS</a> |

## Syntax

**ADJUST\_NCHAR\_COLUMN\_LENGTHS=YES | NO**

### Syntax Description

#### YES

indicates that column lengths are based on the number of characters multiplied by the DBCLIENT\_MAX\_BYTES= value.

#### NO

indicates that column lengths that NCHAR or NVARCHAR columns specify are multiplied by 2.

## Example: No Adjustment for Client-Encoded Column Lengths

NCHAR column lengths are no longer adjusted to client encoding when ADJUST\_NCHAR\_COLUMN\_LENGTHS=NO, as shown in this example.

```
libname x2 &engine &connopt ADJUST_NCHAR_COLUMN_LENGTHS=NO;
proc contents data=x2.char_sem; run;
proc contents data=x2.nchar_sem; run;
proc contents data=x2.byte_sem; run;
proc contents data=x2.mixed_sem; run;
```

In this example, various options have different values.

```
libname x5 &engine &connopt ADJUST_NCHAR_COLUMN_LENGTHS=NO
ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=NO DBCLIENT_MAX_BYTES=3;
proc contents data=x5.char_sem; run;
proc contents data=x5.nchar_sem; run;
proc contents data=x5.byte_sem; run;
proc contents data=x5.mixed_sem; run;
```

This example also uses different values for the various options.

```
libname x6 &engine &connopt ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=YES
ADJUST_NCHAR_COLUMN_LENGTHS=YES DBCLIENT_MAX_BYTES=3;
proc contents data=x6.char_sem; run;
proc contents data=x6.nchar_sem; run;
proc contents data=x6.byte_sem; run;
```

```
proc contents data=x6.mixed_sem; run;
```

## ALLOW\_LARGE\_RESULTS= LIBNAME Statement Option

Specifies whether to allow query results larger than the Google BigQuery maximum response size.

|              |                                                                                                                          |
|--------------|--------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                             |
| Category:    | Data Set Control                                                                                                         |
| Aliases:     | ALR=<br>LARGE_RESULTS=                                                                                                   |
| Default:     | OFF                                                                                                                      |
| Data source: | Google BigQuery                                                                                                          |
| Note:        | Support for this option was added in the April 2021 update for SAS/ACCESS on SAS 9.4M7.                                  |
| See:         | <a href="#">LARGE_RESULTS_DATASET= LIBNAME option</a> ,<br><a href="#">LARGE_RESULTS_EXPIRATION_TIME= LIBNAME option</a> |

## Syntax

**ALLOW\_LARGE\_RESULTS=**[ALWAYS | MUST | OFF](#)

## Required Arguments

### ALWAYS

specifies that all query results are written to a temporary stored table, including query results that are smaller than the maximum response size.

### MUST

specifies that query results that are larger than the maximum response size are written to a temporary stored table.

### OFF

specifies that query results that are larger than the maximum response size trigger an error in the SAS log, and thus the query fails.

## Details

When ALLOW\_LARGE\_RESULTS= is set to ALWAYS or MUST, query results are written to the data set ID that you specify with the LARGE\_RESULTS\_DATASET= LIBNAME option. That data set ID is maintained for the amount of time that you specify with the LARGE\_RESULTS\_EXPIRATION\_TIME= LIBNAME option.

---

## ALLOWED\_SQLCODES= LIBNAME Statement Option

Specifies the SQL warnings and errors to ignore during preparation, execution, and fetch operations.

|              |                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                        |
| Category:    | Data Set Control                                                                                                                                    |
| Default:     | none                                                                                                                                                |
| Requirement: | You must separate multiple values with commas and enclose the entire list in parentheses. Use double quotation marks around each value in the list. |
| Data source: | DB2 under z/OS                                                                                                                                      |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M1.                                                                                             |
| Example:     | <code>libname x db2 allowed_sqlcodes=("144", "-803") ssid=DB2;</code>                                                                               |

---

## Syntax

**ALLOWED\_SQLCODES=(“*sqlcode*”, “*sqlcode*”, “*sqlcode*” ...>>)**

### Syntax Description

#### *sqlcode*

specifies one or more values that you want to control. These values subsequently do not appear in the log.

---

## Details

Use this option to pass a list of specific warnings and errors for the SAS/ACCESS engine to ignore. This is helpful to allow execution to continue for errors or warnings that are for informational purposes only.

---

## ANALYZE= LIBNAME Statement Option

Lets SAS improve performance when a single Hive store table is queried.

|              |                              |
|--------------|------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement |
| Category:    | Data Access                  |
| Default:     | NO                           |
| Data source: | Hadoop                       |

See: ANALYZE= data set option, READ\_METHOD= LIBNAME option, READ\_METHOD= data set option

## Syntax

**ANALYZE=YES | NO**

### Syntax Description

#### YES

specifies that SAS might run an ANALYZE TABLE command to update table statistics. Current table statistics might improve SAS Read performance when a single table is queried. This operation is considered a hint, so if statistics are up-to-date, SAS might not perform the operation. The format of the ANALYZE TABLE command is subject to change in future releases of SAS as needed.

#### NO

specifies that SAS does not perform an additional operation to update table statistics.

## Details

Performance improves when Hive statistics are up-to-date. When the Hive Boolean variable `hive.stats.autogather` is set to TRUE, in most cases Hive automatically gathers statistics. This option can be useful when `hive.stats.autogather` is set to FALSE or when statistics are not being computed. Specifically, Hive statistics are not generated when loading a target table with a file format of TEXT. Users can check the state of Hive statistics using the Hive DESCRIBE FORMATTED command.

## AUTHDOMAIN= LIBNAME Statement Option

Allows connection to a server by specifying the name of an authentication domain metadata object.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Access

Default: none

Restriction: *Hadoop, Impala, Teradata*: When using Kerberos authentication, do not also specify the AUTHDOMAIN= LIBNAME option.

Requirements: For data sources with the SERVER= option, if you specify AUTHDOMAIN=, you must also specify the METASERVER= system option. However, the authentication domain references credentials, so you do not need to explicitly specify database USER= and PASSWORD=. Here is an example:

```
options metauser="metadata-userid" metapass="metadata-password"
           metaport=8561          metaproto=bridge
```

```
metarepository="metadata-repository"  
metaserver="server-name";  
  
libname A1 saphana server=mysrv1 port=30015 authdomain="hanaauth";
```

The authentication domain and the associated login definition must be stored in a metadata repository, and the metadata server must be running to resolve the metadata object specification.

- Interaction: To specify AUTHDOMAIN=, you must also specify SERVER=.
- Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick
- Notes: Support for HAWQ was added in SAS 9.4M3.  
Support for JDBC was added in SAS Viya 3.4.  
Support for Snowflake was added in the August 2019 release of SAS/ACCESS.  
Support for Yellowbrick was added in SAS 9.4M7.
- See: For complete information about creating and using authentication domains, see the credential management topic in *SAS Intelligence Platform: Security Administration Guide*.

---

## Syntax

**AUTHDOMAIN=***authentication-domain*

### Syntax Description

***authentication-domain***

specifies the name of an authentication domain metadata object.

---

**Note:**

Enclose the authentication domain value in quotation marks if the value is specified in lower case or in mixed case.

---

---

## Details

An administrator creates authentication domain definitions while creating a user definition with the User Manager in SAS Management Console. The authentication domain is associated with one or more logs in metadata objects, which provide access to the server. The authentication domain is resolved when the DBMS engine calls the SAS Metadata Server and returns the authentication credentials.

---

## AUTHID= LIBNAME Statement Option

Allows qualified table names with an authorization ID, a user ID, or a group ID.

|              |                                                                                     |
|--------------|-------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                        |
| Category:    | Data Access                                                                         |
| Alias:       | SCHEMA=                                                                             |
| Default:     | none                                                                                |
| Data source: | DB2 under z/OS                                                                      |
| See:         | <a href="#">AUTHID= data set option</a> , <a href="#">DBCONINIT= LIBNAME option</a> |

---

## Syntax

**AUTHID=***authorization-ID*

### Syntax Description

***authorization-ID***

cannot exceed eight characters.

---

## Details

When you specify the AUTHID= option, every table that is referenced by the libref is qualified as *authid.tablename* before any SQL code is passed to the DBMS. If you do not specify a value for AUTHID=, the table name is not qualified before it is passed to the DBMS. After the DBMS receives the table name, it automatically qualifies it with your user ID. You can override the LIBNAME AUTHID= option by using the AUTHID= data set option. This option is not validated until you access a table.

If you specify DBCONINIT="SET CURRENT SQLID='user-ID'", then any value that is specified for AUTHID= is ignored.

---

## AUTOCOMMIT= LIBNAME Statement Option

Indicates whether updates are committed immediately after they are submitted.

|           |                                                                                             |
|-----------|---------------------------------------------------------------------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement                                                                |
| Category: | Data Set Control                                                                            |
| Defaults: | operation-specific [Aster, Greenplum, HAWQ, SAP IQ]<br>data source-specific [ODBC , OLE DB] |

NO [if you are using the SAS/ACCESS LIBNAME statement, the data source provider supports transactions, and the connection is to update data]

NO [for updates and the main LIBNAME connection for Netezza]

NO [DB2 under UNIX and PC Hosts, Microsoft SQL Server, PostgreSQL, SAP HANA, Snowflake, Vertica, Yellowbrick]

YES [for PROC PRINT, Read-only connections, and the SQL pass-through facility]

YES [Amazon Redshift, Informix, MySQL, SAP ASE]

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

See: [AUTOCOMMIT= data set option](#)

## Syntax

**AUTOCOMMIT=YES | NO**

### Syntax Description

#### YES

specifies that all updates, deletes, and inserts are committed (that is, saved to a table) immediately after they are submitted, and no rollback is possible.

#### NO

specifies that the SAS/ACCESS engine automatically performs the commit when it reaches the **DBCOMMIT=** value, or the default number of rows if DBCOMMIT= is not set.

---

#### CAUTION

**For Amazon Redshift, use caution when specifying AUTOCOMMIT=NO with Insert or Update operations.** When AUTOCOMMIT=NO and an error occurs, Amazon Redshift automatically discards any changes that have not been committed. In this situation, the number of inserted rows that is reported might be incorrect.

---

## BL\_ACCOUNTNAME= LIBNAME Statement Option

Specifies the name of the account to use on the external storage system.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading  
Data Access

|              |                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Default:     | none                                                                                                                                                                                       |
| Interaction: | This option is used with the BL_DNSSUFFIX=, BL_FILESYSTEM=, and BL_FOLDER= options to specify the external data location.                                                                  |
| Data source: | Microsoft SQL Server                                                                                                                                                                       |
| Note:        | Support for this option was added in SAS 9.4M7.                                                                                                                                            |
| See:         | <a href="#">BL_ACCOUNTNAME= data set option</a> , <a href="#">BL_DNSSUFFIX= LIBNAME option</a> , <a href="#">BL_FILESYSTEM= LIBNAME option</a> , <a href="#">BL_FOLDER= LIBNAME option</a> |

## Syntax

**BL\_ACCOUNTNAME="account-name"**

### Syntax Description

#### **account-name**

specifies the name of the account to use on the external storage system.

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see [“Bulk Loading to Azure Synapse Analytics” on page 1025](#).

The external data location is generated using this option and the BL\_DNSSUFFIX=, BL\_FILESYSTEM=, and BL\_FOLDER= LIBNAME options. These values are combined to define the URL to an external storage location in an Azure Data Lake Storage Gen2 location:

`https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>`

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

## BL\_APPLICATIONID= LIBNAME Statement Option

Specifies the application ID (a GUID string) for accessing the external storage system.

|              |                                                   |
|--------------|---------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                      |
| Categories:  | Bulk Loading<br>Data Access                       |
| Default:     | none                                              |
| Data source: | Microsoft SQL Server                              |
| Note:        | Support for this option was added in SAS 9.4M7.   |
| See:         | <a href="#">BL_APPLICATIONID= data set option</a> |

Example: bl\_applicationid="b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f"

---

## Syntax

**BL\_APPLICATIONID="string"**

### Syntax Description

**string**

specifies the application ID (a GUID string) for accessing the external storage system.

---

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with a Microsoft Azure Data Lake Storage Gen2 storage account. For more information, “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

---

## BL\_AWS\_CONFIG\_FILE= LIBNAME Statement Option

Specifies the location of the AWS configuration file.

|              |                                                                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                              |
| Categories:  | Bulk Loading<br>Data Access                                                                                                                               |
| Alias:       | BL_AWS_CONFIG=                                                                                                                                            |
| Default:     | ~/.aws/config                                                                                                                                             |
| Requirement: | To specify this option, you must first specify BULKUNLOAD=YES.                                                                                            |
| Data source: | Amazon Redshift                                                                                                                                           |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M6.                                                                                                   |
| See:         | <a href="#">BL_AWS_CONFIG_FILE=</a> data set option, <a href="#">BL_AWS_CREDENTIALS_FILE=</a> data set option, <a href="#">BULKUNLOAD= LIBNAME</a> option |

---

## Syntax

**BL\_AWS\_CONFIG\_FILE=***path-and-file-name*

## Required Argument

***path-and-file-name***

specifies the location of the AWS configuration file. By default, this location is `~/.aws/config`.

# BL\_AWS\_PROFILE\_NAME= LIBNAME Statement Option

Specifies the AWS profile to use when there is more than one profile in the AWS configuration file.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading

Data Access

Alias: BL\_AWS\_PROFILE=

Default: none

Requirement: To specify this option, you must first specify BULKUNLOAD=YES.

Data source: Amazon Redshift

Note: Support for this LIBNAME option was added in SAS 9.4M6.

See: [BULKUNLOAD= LIBNAME option](#)

## Syntax

**BL\_AWS\_PROFILE\_NAME=*profile-name***

## Required Argument

***profile-name***

specifies the profile to use if there is more than one profile in the AWS configuration file.

## Details

If you specify more than one profile in your AWS configuration file, then each profile has a name that is specified in square brackets. Here is a sample configuration file with two profiles, default and analyst.

```
[default]
region=us-west-2
output=text
```

```
[analyst]
```

```
region=us-east-1  
output=json
```

To use the analyst profile, specify the following code.

```
data myclass (bulkunload=yes  
             bl_bucket='myBucket/'  
             bl_aws_profile_name=analyst  
             );  
set sashelp.class;  
run;
```

---

## BL\_BUCKET= LIBNAME Statement Option

Specifies the bucket name to use when transferring data files in bulk.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading

Data Access

Default: none

Requirements: To specify this option, you must first specify BULKUNLOAD=YES.

The bucket name that you specify must already exist.

Interaction: When BULKUNLOAD=YES, the BL\_BUCKET= LIBNAME option is required.

Data source: Amazon Redshift, Google BigQuery

Notes: Support for this LIBNAME option was added in SAS 9.4M6.

Support for Google BigQuery was added in SAS 9.4M7.

---

## Syntax

**BL\_BUCKET='bucket-name'**

---

## Details

For Amazon Redshift, BL\_BUCKET= refers to an Amazon S3 bucket. For more information, see [Amazon Web Services S3 information](#).

For Google BigQuery, BL\_BUCKET= refers to a Google Cloud Storage bucket.

---

## BL\_COMPRESS= LIBNAME Statement Option

Specifies whether to compress data files using the gzip format.

Valid in: SAS/ACCESS LIBNAME statement

|              |                                                                                                                     |
|--------------|---------------------------------------------------------------------------------------------------------------------|
| Categories:  | Bulk Loading<br>Data Set Control                                                                                    |
| Default:     | NO                                                                                                                  |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES or BULKUNLOAD=YES.                                      |
| Data source: | Amazon Redshift, Microsoft SQL Server                                                                               |
| Notes:       | Support for this LIBNAME option was added in SAS 9.4M6.<br>Support for Microsoft SQL Server was added in SAS 9.4M7. |
| See:         | <a href="#">BULKUNLOAD= LIBNAME option</a>                                                                          |

## Syntax

**BL\_COMPRESS=YES | NO**

## Details

If you have a slow network or very large data files, using BL\_COMPRESS=YES might improve performance.

## BL\_CONFIG= LIBNAME Statement Option

Specifies the location of the optional configuration file that the SAS communication layer uses to interact with the Amazon S3 tool.

|              |                                                                                              |
|--------------|----------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                     |
| Categories:  | Bulk Loading<br>Data Access                                                                  |
| Alias:       | BL_CONFIG_FILE=                                                                              |
| Default:     | none                                                                                         |
| Requirement: | To specify this option, you must first specify BULKUNLOAD=YES.                               |
| Data source: | Amazon Redshift                                                                              |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M6.                                      |
| See:         | <a href="#">BULKUNLOAD= LIBNAME option</a>                                                   |
| Examples:    | <pre>BL_CONFIG='c:\temp\'</pre> <p>[Windows]</p> <pre>BL_CONFIG='/temp/'</pre> <p>[UNIX]</p> |

## Syntax

**BL\_CONFIG='<path-and-file-name>'**

### Required Argument

***path-and-file-name***

specifies the host-specific directory path where the optional configuration file for communications with S3 is located.

---

## Details

The configuration file that you specify can contain options for communication with the Amazon S3 tool. The file format uses name=value syntax with one option per line. For example, you can specify the following S3 options in this file:

```
ssl=yes
keyId=access-key-ID
secret=secret-access-key
region=aws-region
sessionToken=temporary-token
```

For more information, see [information about credentials](#) on the Amazon Web Services website.

---

## BL\_DEFAULT\_DIR= LIBNAME Statement Option

Specifies where bulk loading creates all intermediate files.

|               |                                                                                                                                                                                                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                  |
| Categories:   | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                              |
| Defaults:     | temporary file directory that is specified by the UTILLOC= system option [Amazon Redshift, Aster, Google BigQuery, Microsoft SQL Server, MySQL]<br><i>database-name</i> [Oracle]                                                                                              |
| Restriction:  | The value that you specify must not contain a space.                                                                                                                                                                                                                          |
| Requirements: | To specify this option, you must first set the BULKLOAD= LIBNAME option or the BULKLOAD= data set option to YES. Amazon Redshift supports only the BULKLOAD= data set option.<br><br>This option must end with a backslash on Windows or a forward slash on UNIX—for example: |
|               | <code>BL_DEFAULT_DIR='c:\temp\'</code><br><code>BL_DEFAULT_DIR='/temp/'</code>                                                                                                                                                                                                |
| Supports:     | CTL, DAT, LOG, BAD, and DSC intermediate bulk-load files (Oracle)                                                                                                                                                                                                             |
| Data source:  | Amazon Redshift, Aster, Google BigQuery, Microsoft SQL Server, MySQL, Oracle                                                                                                                                                                                                  |

- Notes:      Support for Amazon Redshift was added in SAS 9.4M6.  
               Support for Google BigQuery was added in the August 2019 release of SAS/ACCESS.  
               Support for Microsoft SQL Server was added in SAS 9.4M7.
- See:        [BULKLOAD= LIBNAME option](#), [BL\\_DEFAULT\\_DIR= data set option](#)

## Syntax

**BL\_DEFAULT\_DIR='host-specific-directory-path'**

### Required Argument

***host-specific-directory-path***

specifies the host-specific directory path where intermediate bulk-load files are created.

## Details

The value that you specify for this option is prepended to the file name. Be sure to provide the complete, host-specific directory path that includes the file and directory separator character to accommodate all platforms.

This value overrides the default location for interim files that is specified by the UTILLOC= system option.

*Microsoft SQL Server:* This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with a Microsoft Azure Data Lake Storage Gen2 storage account. For more information, “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

## Example: Create All Files in a Temporary Directory

In this example, bulk loading Oracle data creates all related files in the c:\temp directory:

```
libname mypath oracle user=myusr1 password=mypwd1
    bl_default_dir='c:\temp\';
```

---

# BL\_DELETE\_DATAFILE= LIBNAME Statement Option

Specifies whether to delete only the data file or all files that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

|              |                                                                                                                                                                                                                                                                   |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                      |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                  |
| Default:     | YES                                                                                                                                                                                                                                                               |
| Requirement: | To specify this option, you must also set BULKLOAD=YES or BULKUNLOAD=YES.                                                                                                                                                                                         |
| Data source: | Amazon Redshift, Google BigQuery, Microsoft SQL Server                                                                                                                                                                                                            |
| Notes:       | Support for this option was added in SAS 9.4M6.<br>Support for Google BigQuery was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7.                                                                    |
| See:         | <a href="#">BL_DATAFILE= data set option</a> , <a href="#">BL_DELETE_DATAFILE= data set option</a> ,<br><a href="#">BL_DELETE_ONLY_DATAFILE= data set option</a> , <a href="#">BL_USE_MANIFEST data set option</a> ,<br><a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_DELETE\_DATAFILE=YES | NO**

### Syntax Description

#### **YES**

deletes all (data, control, and log) files that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

#### **NO**

does not delete these files.

---

## Details

*Amazon Redshift:* Setting BL\_DELETE\_DATAFILES=YES deletes data files that are created during the bulk retrieval process. If BL\_USE\_MANIFEST=YES, then manifest files are deleted as well. Files are deleted from the local machine and from the S3 bucket.

*Microsoft SQL Server:* This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

## BL\_DELIMITER= LIBNAME Statement Option

Specifies override of the default delimiter character for separating columns of data during bulk loading or unloading (retrieval).

|              |                                                                                                                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                   |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                               |
| Default:     | the bell character (ASCII 0x07)                                                                                                                                                                |
| Requirement: | To specify this option, you must first set the BULKUNLOAD= data set option or LIBNAME option to YES.                                                                                           |
| Data source: | Amazon Redshift, Google BigQuery, Microsoft SQL Server                                                                                                                                         |
| Notes:       | Support for this option was added in SAS 9.4M6.<br>Support for Google BigQuery was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7. |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">BULKUNLOAD= LIBNAME option</a> , <a href="#">BULKUNLOAD= data set option</a>                                                           |

## Syntax

**BL\_DELIMITER='<any-single-character>'**

## Details

Here is when you might want to use this option:

- to override the default delimiter character that the interface uses to separate columns of data that are transferred to or retrieved from the DBMS during bulk loading or retrieval.
- if your character data contains the default delimiter character, to avoid any problems while parsing the data stream.

*Amazon Redshift:* Only Amazon Redshift supports bulk retrieval of data into SAS.

## BL\_DNSSUFFIX= LIBNAME Statement Option

Specifies the network host name where the storage system resides.

|             |                              |
|-------------|------------------------------|
| Valid in:   | SAS/ACCESS LIBNAME statement |
| Categories: | Bulk Loading<br>Data Access  |

|              |                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Default:     | dfs.core.windows.net                                                                                                                                                                       |
| Interaction: | This option is used with the BL_ACCOUNTNAME=, BL_FILESYSTEM=, and BL_FOLDER= options to specify the external data location.                                                                |
| Data source: | Microsoft SQL Server                                                                                                                                                                       |
| Note:        | Support for this option was added in SAS 9.4M7.                                                                                                                                            |
| See:         | <a href="#">BL_ACCOUNTNAME= LIBNAME option</a> , <a href="#">BL_DNSSUFFIX= data set option</a> , <a href="#">BL_FILESYSTEM= LIBNAME option</a> , <a href="#">BL_FOLDER= LIBNAME option</a> |

## Syntax

**BL\_DNSSUFFIX="***network-storage-host-name***"**

### Syntax Description

#### ***network-storage-host-name***

specifies the network host name where the storage system resides. It is referred to as a suffix because the value is typically appended to the end of the storage account name.

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see [“Bulk Loading to Azure Synapse Analytics” on page 1025](#).

The external data location is generated using this option and the BL\_ACCOUNTNAME=, BL\_FILESYSTEM=, and BL\_FOLDER= LIBNAME options. These values are combined to define the URL to access an Azure Data Lake Storage Gen2 location:

`https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>`

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

## BL\_ENCKEY= LIBNAME Statement Option

Specifies the name of the encryption key to use when you access an Amazon S3 environment.

|             |                                   |
|-------------|-----------------------------------|
| Valid in:   | SAS/ACCESS LIBNAME statement      |
| Categories: | Bulk Loading<br>Data Access       |
| Aliases:    | BL_ENC_KEY=<br>BL_ENCRYPTION_KEY= |
| Default:    | none                              |

|              |                                                                                                                                                                                 |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | Amazon Redshift                                                                                                                                                                 |
| Note:        | Support for this option was added in SAS 9.4M6.                                                                                                                                 |
| See:         | <a href="#">BL_ENCKEY= data set option</a> , <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKUNLOAD= LIBNAME option</a> , <a href="#">S3 procedure documentation</a> |

## Syntax

**BL\_ENCKEY=***encryption-key-name*

### Syntax Description

#### ***encryption-key-name***

specifies the name that is associated with an encryption key. The name matches a name that was assigned to an encryption key by using the S3 procedure.

## Details

When you specify the BL\_ENCKEY= option, you enable server-side encryption for all communication with an Amazon S3 environment.

You use the S3 procedure to register an encryption key and assign a corresponding name. This is the name that you provide as the value for the BL\_ENCKEY= option. For more information, see the ENCKEY statement of the “[S3 Procedure](#)” in *Base SAS Procedures Guide*.

This option works well with the BULKLOAD= and BULKUNLOAD= options.

## BL\_FILESYSTEM= LIBNAME Statement Option

Specifies the name of the file system within the external storage system.

|              |                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                               |
| Categories:  | Bulk Loading<br>Data Access                                                                                                                                                                |
| Default:     | none                                                                                                                                                                                       |
| Interaction: | This option is used with the BL_ACCOUNTNAME=, BL_DNSSUFFIX=, and BL_FOLDER= options to specify the external data location.                                                                 |
| Data source: | Microsoft SQL Server                                                                                                                                                                       |
| Note:        | Support for this option was added in SAS 9.4M7.                                                                                                                                            |
| See:         | <a href="#">BL_ACCOUNTNAME= LIBNAME option</a> , <a href="#">BL_DNSSUFFIX= LIBNAME option</a> , <a href="#">BL_FILESYSTEM= data set option</a> , <a href="#">BL_FOLDER= LIBNAME option</a> |

---

## Syntax

**BL\_FILESYSTEM="file-system-name"**

### Syntax Description

***file-system-name***

specifies the name of the file system within the external storage system that files are to be read from and written to.

---

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

The external data location is generated using this option and the BL\_ACCOUNTNAME=, BL\_DNSSUFFIX=, and BL\_FOLDER= LIBNAME options. These values are combined to define the URL to access an Azure Data Lake Storage Gen2 location:

`https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>`

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

---

## BL\_FOLDER= LIBNAME Statement Option

Specifies the folder or file path for the data in the external storage system.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading  
Data Access

Default: none

Interaction: This option is used with the BL\_ACCOUNTNAME=, BL\_DNSSUFFIX=, and BL\_FILESYSTEM= options to specify the external data location.

Data source: Microsoft SQL Server

Note: Support for this option was added in SAS 9.4M7.

See: [BL\\_ACCOUNTNAME= LIBNAME option](#), [BL\\_DNSSUFFIX= LIBNAME option](#), [BL\\_FILESYSTEM= LIBNAME option](#), [BL\\_FOLDER= data set option](#)

---

## Syntax

**BL\_FOLDER="file-path-and-folder"**

## Syntax Description

### **file-path-and-folder**

specifies the folder or file path for the data in the external storage system. The location path begins with the container name.

---

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

The external data location is generated using this option and the **BL\_ACCOUNTNAME=**, **BL\_DNSSUFFIX=**, and **BL\_FILESYSTEM=** LIBNAME options. These values are combined to define the URL to access an Azure Data Lake Storage Gen2 location:

`https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>`

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

---

## BL\_FORMAT= LIBNAME Statement Option

Specifies the format of the Hadoop bulk load staging file.

|              |                                                                                           |
|--------------|-------------------------------------------------------------------------------------------|
| Category:    | Performance                                                                               |
| Default:     | TEXT                                                                                      |
| Restriction: | This option is available only on Linux.                                                   |
| Requirement: | To specify this option, you must specify BULKLOAD=YES. BULKLOAD=YES is the default value. |
| Data source: | Hadoop                                                                                    |
| Note:        | Support for this option starts in <a href="#">SAS 9.4M8</a> .                             |

---

## Syntax

**BL\_FORMAT=TEXT | ORC**

## Syntax Description

### **TEXT**

specifies plain text format.

### **ORC**

specifies the Apache ORC (Optimized Row Columnar) open-source file format.

## Details

By default, the Hadoop engine creates its bulk load staging file as a delimited text file. The ORC format can offer advantages in size, performance, and precision because data values do not need to be textualized in order to be loaded.

## See Also

### **Data Set Options:**

- “[BL\\_FORMAT= Data Set Option](#)” on page 441

---

## BL\_HOST= LIBNAME Statement Option

Specifies the WebHDFS host name or IP address of the server where the external data file is stored.

|              |                                                              |
|--------------|--------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                 |
| Categories:  | Bulk Loading<br>Data Access                                  |
| Alias:       | BLHOST=, BULKLOAD_HOST=                                      |
| Default:     | SERVER= value, if SERVER= is set; otherwise, none.           |
| Restriction: | Enclose the name in quotation marks.                         |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES. |
| Data source: | Impala                                                       |
| Note:        | Support for Impala was added in SAS 9.4M2.                   |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a>                     |

---

## Syntax

**BL\_HOST='host-name'**

### Syntax Description

#### ***host-name***

specifies the name or IP address of the server where the external data file is stored.

## Details

When you are bulk loading data, use this option to specify the name or IP address of the server where external data is stored. By default, this value is the same as the value of SERVER=, when the SERVER= option is specified.

---

## BL\_IDENTITY= LIBNAME Statement Option

Specifies the authentication method for the COPY command.

|              |                                                                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                       |
| Categories:  | Bulk Loading<br>Data Access                                                                                                                        |
| Default:     | none                                                                                                                                               |
| Interaction: | The value for this option and the BL_SECRET= value are used for the CREDENTIAL argument with the COPY command in Azure Synapse Analytics (SQL DW). |
| Data source: | Microsoft SQL Server                                                                                                                               |
| Note:        | Support for this option was added in SAS 9.4M7.                                                                                                    |
| See:         | <a href="#">BL_IDENTITY= data set option</a> , <a href="#">BL_SECRET= LIBNAME option</a>                                                           |
| Example:     | bl_identity='Shared Access Signature'                                                                                                              |

---

## Syntax

**BL\_IDENTITY="***identity-value***"**

### Syntax Description

#### *identity-value*

specifies the authentication method for the COPY command in Azure Synapse Analytics (SQL DW).

---

## Details

The BL\_IDENTITY= value is used as part of the CREDENTIAL argument that you specify for the COPY command in Azure Synapse Analytics (SQL DW). This functionality is supported when you are using an Azure Data Lake Gen2 account.

---

## BL\_KEEPIDENTITY= LIBNAME Statement Option

Determines whether the identity column that is created during bulk loading is populated with values that Microsoft SQL Server generates or with values that the user provides.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading  
Data Set Control

Default: NO

Restriction: This option is valid only when you use the Microsoft SQL Server provider.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: OLE DB

See: [BL\\_KEEPIDENTITY= data set option, BULKLOAD= LIBNAME option](#)

---

## Syntax

**BL\_KEEPIDENTITY=YES | NO**

### Syntax Description

#### **YES**

specifies that the user must provide values for the identity column.

#### **NO**

specifies that Microsoft SQL Server generates values for an identity column in the table.

---

## BL\_KEEPNULLS= LIBNAME Statement Option

Indicates how NULL values in Microsoft SQL Server columns that accept NULL are handled during bulk loading.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading  
Data Set Control

Default: YES

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: OLE DB

See: [BL\\_KEEPNULLS= data set option, BULKLOAD= LIBNAME option](#)

---

## Syntax

**BL\_KEEPNULLS=YES | NO**

### Syntax Description

#### YES

specifies that Microsoft SQL Server preserves NULL values that the OLE DB interface inserts.

#### NO

specifies that Microsoft SQL Server replaces NULL values that the OLE DB interface inserts with a default value, as specified in the DEFAULT constraint.

---

## Details

This option affects values in only Microsoft SQL Server columns that accept NULL and have a DEFAULT constraint.

---

## BL\_KEY= LIBNAME Statement Option

Specifies the Amazon Web Services access key that is used with key-based access control.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading  
Data Access

Default: none

Requirement: To specify this option, you must first specify BULKUNLOAD=YES.

Data source: Amazon Redshift

Note: Support for this LIBNAME option was added in SAS 9.4M6.

Tip: If you are using temporary token credentials, this is the temporary access key ID.

See: [BULKUNLOAD= LIBNAME option](#)

---

## Syntax

**BL\_KEY=*key-value***

### Required Argument

#### *key-value*

specifies an AWS key that you use to access the AWS environment.

---

## Details

For more information about managing access keys, see the [Amazon Web Services \(AWS\) documentation](#).

---

## BL\_LOG= LIBNAME Statement Option

Specifies the name of the error file where all errors are written when BULKLOAD=YES.

|              |                                                                                                |
|--------------|------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                   |
| Categories:  | Bulk Loading<br>Data Set Control                                                               |
| Alias:       | BCP_ERRORFILE=                                                                                 |
| Default:     | none                                                                                           |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                   |
| Data source: | Microsoft SQL Server, ODBC                                                                     |
| Note:        | Support for Microsoft SQL Server was added in SAS 9.4M7.                                       |
| Tip:         | If you do not specify BL_LOG=, then errors are not recorded during bulk loading.               |
| See:         | <a href="#">BL_LOG= data set option on page 455</a> , <a href="#">BULKLOAD= LIBNAME option</a> |

---

## Syntax

**BL\_LOG=<'>*file-name*<'>**

### Required Argument

***file-name***  
specifies the name of the designated log file.

---

## Details

**Microsoft SQL Server:** This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account.

---

## BL\_MAXERRORS= LIBNAME Statement Option

Specifies the maximum number of rejected rows that are allowed in the load before the COPY command is canceled.

|              |                                                           |
|--------------|-----------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                              |
| Category:    | Bulk Loading                                              |
| Default:     | 0                                                         |
| Data source: | Microsoft SQL Server                                      |
| Note:        | Support for this option was added in SAS 9.4M7.           |
| See:         | <a href="#">BL_MAXERRORS= data set option on page 458</a> |

## Syntax

**BL\_MAXERRORS=***value*

### Syntax Description

#### *value*

specifies the maximum number of rejected rows that are allowed in the load before the COPY command is canceled.

## Details

This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account.

## BL\_NUM\_READ\_THREADS= LIBNAME Statement Option

Specifies the number of threads to use for bulk unloading from the DBMS into SAS.

|              |                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                    |
| Categories:  | Bulk Loading<br>Data Access                                                                                                                     |
| Default:     | 4                                                                                                                                               |
| Data source: | Amazon Redshift, Google BigQuery                                                                                                                |
| Notes:       | Support for this LIBNAME option was added in SAS 9.4M6.<br>Support for Google BigQuery was added in SAS 9.4M7.                                  |
| See:         | <a href="#">BL_NUM_READ_THREADS= data set option</a> , <a href="#">BULKUNLOAD= data set option</a> , <a href="#">BULKUNLOAD= LIBNAME option</a> |

## Syntax

**BL\_NUM\_READ\_THREADS=***value*

### Syntax Description

***value***

specifies the number of threads that are used to perform bulk unloading (retrieval) of data from a DBMS.

## BL\_OPTIONS= LIBNAME Statement Option

Passes to the DBMS bulk-load facility options that affect how it loads and processes data.

|               |                                                                                                                                                                                   |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                      |
| Categories:   | Bulk Loading<br>Data Set Control                                                                                                                                                  |
| Alias:        | BCP_OPTIONS= [Microsoft SQL Server, ODBC]                                                                                                                                         |
| Default:      | none                                                                                                                                                                              |
| Restriction:  | <i>ODBC, OLE DB</i> : This option is valid only when you use the Microsoft SQL Server driver or the Microsoft SQL Server provider on Windows platforms.                           |
| Requirements: | <i>Amazon Redshift</i> : To specify this option, you must first set BULKUNLOAD=YES.<br><i>ODBC, OLE DB</i> : To specify this option, you must first set BULKLOAD=YES.             |
| Data source:  | Amazon Redshift, Microsoft SQL Server, ODBC, OLE DB                                                                                                                               |
| Note:         | Support for Microsoft SQL Server was added in SAS 9.4M7.                                                                                                                          |
| See:          | <b>BL_OPTIONS=</b> data set option, <b>BULKLOAD=</b> LIBNAME option, <b>BULKLOAD=</b> data set option, <b>BULKUNLOAD=</b> LIBNAME option, <b>UPDATE_LOCK_TYPE=</b> LIBNAME option |

## Syntax

**BL\_OPTIONS='***option-1 <, option-2...>***'**

### Required Argument

***option***

specifies a bulk-load option that you want to set.

## Details

You can use BL\_OPTIONS= to pass options to the DBMS bulk-load facility when it is called, thereby affecting how data is loaded and processed. You must enclose the entire string of one or more options in quotation marks. Separate multiple options with commas.

*Microsoft SQL Server:* This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account. Options should be specified as they are used by the WITH clause in the COPY INTO command for Azure Synapse Analytics (SQL DW). For example, you can specify the Azure FIELDTERMINATOR option within the BL\_OPTIONS= string instead of specifying the BL\_DELIMITER= LIBNAME option.

By default, no options are specified. This option takes the same values as the *-h* HINT option of the Microsoft BCP utility. See the Microsoft SQL Server documentation for more information about bulk copy options.

*ODBC:* Supported hints are ORDER, ROWS\_PER\_BATCH, KILOBYTES\_PER\_BATCH, TABLOCK, and CHECK\_CONSTRAINTS. If you specify [UPDATE\\_LOCK\\_TYPE=TABLE](#), the TABLOCK hint is automatically added.

---

## BL\_PORT= LIBNAME Statement Option

Specifies the port number for writing table data to the Hadoop cluster.

|              |                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                    |
| Categories:  | Bulk Loading<br>Data Access                                                                                     |
| Alias:       | BLPORT=, BULKLOAD_PORT=                                                                                         |
| Defaults:    | 8020 [Hadoop]<br>50070 [Impala]                                                                                 |
| Restriction: | This option is required if Hadoop HDFS service is running on a port other than 8020 (Hadoop) or 50070 (Impala). |
| Data source: | Hadoop, Impala                                                                                                  |
| Note:        | Support for Impala was added in SAS 9.4M2.                                                                      |
| Tip:         | You can use this option without setting <a href="#">BULKLOAD=YES</a> .                                          |
| See:         | <a href="#">PORT=</a> connection option for <a href="#">Hadoop</a> or <a href="#">Impala</a>                    |

---

## Syntax

**BL\_PORT=***port*

## Syntax Description

### *port*

specifies the port number to use to write table data.

---

## Details

Use this option to specify only the port number that bulk loading uses to write data to the Hadoop cluster using the HDFS streaming server.

---

## BL\_RECOVERABLE= LIBNAME Statement Option

Determines whether the LOAD process is recoverable.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading

Data Set Control

Default: NO

Restriction: This option applies only when you use a CREATE TABLE AS SELECT statement with the LOAD FROM CURSOR method.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Interaction: To view log information related to this option, specify a log file for the BL\_LOG= option.

Data source: DB2 under UNIX and PC Hosts

Note: Support for this option was added in SAS Viya 3.5.

See: [BL\\_LOG= LIBNAME option](#), [BL\\_RECOVERABLE= data set option](#), [BULKLOAD= LIBNAME option](#), [SASTRACE= system option](#)

---

## Syntax

**BL\_RECOVERABLE=YES | NO**

## Syntax Description

### **YES**

specifies that the LOAD process is recoverable.

### **NO**

specifies that the LOAD process is not recoverable.

## Details

This LIBNAME option applies only in the case where you are creating a table in a DB2 location from a table that resides in a DB2 location. In this case, a CREATE TABLE AS SELECT statement is passed to the database for processing. To see the SQL code that is generated, specify the SASTRACE= system option.

---

## BL\_REGION= LIBNAME Statement Option

Specifies the Amazon Web Services (AWS) region from which data is being retrieved.

|              |                                                                |
|--------------|----------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                   |
| Categories:  | Bulk Loading<br>Data Access                                    |
| Default:     | none                                                           |
| Requirement: | To specify this option, you must first specify BULKUNLOAD=YES. |
| Data source: | Amazon Redshift                                                |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M6.        |
| See:         | <a href="#">BULKUNLOAD= LIBNAME option</a>                     |

---

## Syntax

**BL\_REGION=***region*

### Required Argument

*region*

specifies the AWS region from which S3 data is being loaded. You must specify the region when using the S3 tool to load data.

---

## Details

For more information about AWS regions, see the [COPY Parameter Reference](#).

---

## BL\_SECRET= LIBNAME Statement Option

Specifies the secret access key that is used when you transfer data in bulk.

|             |                              |
|-------------|------------------------------|
| Valid in:   | SAS/ACCESS LIBNAME statement |
| Categories: | Bulk Loading                 |

|              |                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | Data Access                                                                                                                                                                       |
| Default:     | none                                                                                                                                                                              |
| Requirement: | To specify this option, you must also specify BULKLOAD=YES or BULKUNLOAD=YES.                                                                                                     |
| Interaction: | <i>Microsoft SQL Server:</i> The value for this option and the BL_IDENTITY= value are used for the CREDENTIAL argument with the COPY command in Azure Synapse Analytics (SQL DW). |
| Data source: | Amazon Redshift, Microsoft SQL Server                                                                                                                                             |
| Notes:       | Support for this LIBNAME option was added in SAS 9.4M6.<br>Support for Microsoft SQL Server was added in SAS 9.4M7.                                                               |
| Tip:         | To increase security, you can encode the key value by using PROC PWENCODE.                                                                                                        |
| See:         | <a href="#">BL_IDENTITY= LIBNAME option</a> , <a href="#">BL_KEY= LIBNAME option</a> , <a href="#">BL_SECRET= data set option</a> , <a href="#">BULKUNLOAD= LIBNAME option</a>    |

## Syntax

**BL\_SECRET=***secret-access-key*

### Required Argument

#### *secret-access-key*

specifies the secret access key value to access a data source.

## Details

If you use PROC PWENCODE to encode the secret, SAS/ACCESS decodes the value before passing it on to the data source.

*Amazon Redshift:* This value is the Amazon Web Services (AWS) secret access key or a temporary secret access key. An AWS secret access key is associated with the key ID that you specified with the BL\_KEY= LIBNAME option or data set option. If you are using temporary token credentials, this value is the temporary secret access key.

*Microsoft SQL Server:* This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account. The *secret-access-key* specifies the secret value for the CREDENTIAL argument of the COPY command.

## BL\_TIMEOUT= LIBNAME Statement Option

Specifies the maximum completion time for a storage task, in seconds.

Valid in: SAS/ACCESS LIBNAME statement

Category: Bulk Loading

Default: 60

Data source: Microsoft SQL Server

Note: Support for this option was added in SAS 9.4M7.

See: [BL\\_TIMEOUT= data set option](#)

---

## Syntax

**BL\_TIMEOUT=***value*

### Syntax Description

#### **value**

specifies the maximum completion time for a storage task, expressed as a number of seconds. If the task does not complete within the specified time, the storage system might return an error.

---

## Details

This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account.

---

## BL\_TOKEN= LIBNAME Statement Option

Specifies a temporary token that is associated with the temporary credentials that you specify with the BL\_KEY= and BL\_SECRET= options.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading  
Data Access

Default: none

Requirement: To specify this option, you must first specify BULKUNLOAD=YES.

Data source: Amazon Redshift

Notes: You can safely ignore any notes you receive about your token being too long.  
Support for this LIBNAME option was added in SAS 9.4M6.

See: [BL\\_KEY= LIBNAME option](#), [BL\\_SECRET= LIBNAME option](#), [BULKUNLOAD= LIBNAME option](#)

---

## Syntax

**BL\_TOKEN=***temporary-token*

## Required Argument

### **temporary-token**

specifies a temporary token that is associated with the temporary credentials that you provide with BL\_KEY= and BL\_SECRET= LIBNAME options.

---

## Details

When you obtain the temporary credentials that you use to access the Amazon Web Services environment, you obtain an AWS key, an AWS secret key, and an AWS token. These credentials are valid only for a limited amount of time.

---

## BL\_USE\_ESCAPE= LIBNAME Statement Option

Specifies whether to escape a predefined set of special characters during bulk loading or bulk retrieval.

Valid in: SAS/ACCESS LIBNAME statement

Categories: Bulk Loading

Data Set Control

Default: NO

Data source: Amazon Redshift, Microsoft SQL Server

Notes: Support for this LIBNAME option was added in SAS 9.4M6.

Support for Microsoft SQL Server was added in SAS 9.4M7.

See: [BL\\_DELIMITER= LIBNAME option](#), [BL\\_USE\\_ESCAPE= data set option](#)

---

## Syntax

**BL\_USE\_ESCAPE=YES | NO**

## Required Arguments

### **YES**

specifies that occurrences of a backslash ('\'), newline ('\n'), carriage return ('\r'), or user-defined delimiter receives a backslash inserted before it. The backslash preserves occurrences of these characters in the data that is transferred during bulk loading or bulk retrieval. Any occurrence of the NULL character (0x00) is replaced by a space (' ').

### **NO**

specifies that occurrences of a backslash ('\'), newline ('\n'), carriage return ('\r'), or user-defined delimiter are not escaped in the data that is transferred during bulk loading or bulk retrieval.

## Details

The predefined special characters that are treated as escape characters include the backslash ('\'), newline ('\n'), and return ('\r'). If you have specified another character for the BL\_DELIMITER= LIBNAME or data set option, that value is also treated as an escape character.

*Amazon Redshift:* A bulk-loading or bulk-unloading operation that is executed in Amazon Redshift uses the COPY command.

*Microsoft SQL Server:* This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account. Bulk loading to Azure Synapse Analytics uses the COPY command.

---

## BL\_USE\_LOG= LIBNAME Statement Option

Specifies whether to get reports on rejected rows and the corresponding error file.

|              |                                                                                      |
|--------------|--------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                         |
| Category:    | Bulk Loading                                                                         |
| Default:     | NO                                                                                   |
| Data source: | Microsoft SQL Server                                                                 |
| Note:        | Support for this option was added in SAS 9.4M7.                                      |
| See:         | <a href="#">BL_LOG= LIBNAME option</a> , <a href="#">BL_USE_LOG= data set option</a> |

---

## Syntax

**BL\_USE\_LOG=YES | NO**

### Required Arguments

#### YES

specifies that a log that contains rejected rows and corresponding errors should be created during bulk loading. The log is stored in the location that is specified by the BL\_LOG= option.

#### NO

specifies not to create a bulk loading log.

---

## Details

If no log location is specified for BL\_LOG=, then the bulk loading log is stored in the temporary location that is specified by BL\_DEFAULT\_DIR=. Make sure that you have permission to write to the log location.

This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account. To obtain Write permission for an Azure Synapse Analytics location, you might need to obtain a Shared Access Signature. For more information, see your Azure documentation.

---

## BL\_USE\_SSL= LIBNAME Statement Option

Specifies whether to use TLS encryption for connections to Amazon S3.

|              |                                                                |
|--------------|----------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                   |
| Categories:  | Bulk Loading<br>Data Set Control                               |
| Default:     | YES                                                            |
| Requirement: | To specify this option, you must first specify BULKUNLOAD=YES. |
| Data source: | Amazon Redshift                                                |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M6.        |
| See:         | <a href="#">BULKUNLOAD= LIBNAME option</a>                     |

---

## Syntax

**BL\_USE\_SSL=YES | NO**

---

## BULKLOAD= LIBNAME Statement Option

Determines whether SAS uses a DBMS facility to insert data into a DBMS table.

|              |                                                                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                           |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                       |
| Aliases:     | BCP= [ODBC]<br>FASTLOAD= [Teradata]                                                                                                                                                                                    |
| Defaults:    | YES [Hadoop, Impala, JDBC, Spark]<br>NO [Aster, Google BigQuery, Greenplum, HAWQ, Microsoft SQL Server, MySQL, ODBC, OLE DB, Teradata, Yellowbrick]<br>none [Snowflake]                                                |
| Data source: | Aster, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, MySQL, ODBC, OLE DB, Snowflake, Spark, Teradata, Yellowbrick                                                                      |
| Notes:       | Support for Impala was added in SAS 9.4M2.<br>Support for Greenplum, HAWQ, JDBC, and MySQL was added in SAS Viya 3.4.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS. |

Support for Microsoft SQL Server was added in SAS 9.4M7.

Support for Spark was added in SAS 9.4M7.

Support for Yellowbrick was added in SAS 9.4M7.

See: [BULKUNLOAD= LIBNAME option](#), [BULKLOAD= data set option](#), [BULKUNLOAD= data set option](#), “[Bulk Loading for Impala](#)”

## Syntax

**BULKLOAD=YES | NO**

### Syntax Description

#### YES

calls a DBMS-specific bulk-load facility to insert or append rows to a DBMS table.

#### NO

does not call the DBMS bulk-load facility.

## Details

*Hadoop and Spark:* When BULKLOAD=YES, the engine uses HDFS to load data into Hive. HDFS connectivity is therefore required, except for operations that occur when DBDIRECTEXEC is enabled. When BULKLOAD=NO, you can write to Hive without HDFS connectivity.

*Hadoop and Impala:* Set your environment variables appropriately before you specify BULKLOAD=YES. If you use SAS and Hadoop JAR files to interact directly with your HDFS system, then specify the SAS\_HADOOP\_CONFIG\_PATH and SAS\_HADOOP\_JAR\_PATH environment variables. If you use the WebHDFS interface to push data to HDFS, then specify SAS\_HADOOP\_CONFIG\_PATH and specify SAS\_HADOOP\_RESTFUL=1. Spark does not require SAS\_HADOOP\_RESTFUL.

For Impala, specifying BULKLOAD=NO does not change the loading behavior since all data is loaded in bulk. However, specifying BULKLOAD=YES is required before you can specify other options that are related to bulk loading.

For more information, see [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#).

*JDBC:* When BULKLOAD=YES, JDBC uses BatchUpdate capabilities to load data. If BULKLOAD=NO or if the JDBC driver does not support BatchUpdate, then SAS/ACCESS Interface to JDBC uses SQL to load data.

*Microsoft SQL Server:* The BULKLOAD= LIBNAME option enables Microsoft SQL Server to load data to Azure Synapse Analytics (SQL DW) with a Microsoft Azure Data Lake Storage Gen2 storage account. For more information, see [“Bulk Loading to Azure Synapse Analytics” on page 1025](#).

*Yellowbrick:* This option calls the YBLOAD utility.

---

## BULKUNLOAD= LIBNAME Statement Option

Rapidly retrieves (fetches) a large number of rows from an external data set.

|              |                                                                                                                                                                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                            |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                        |
| Default:     | NO                                                                                                                                                                                                                                                                                      |
| Restriction: | To specify BULKUNLOAD=YES for Netezza, you must be the Netezza admin user, or you must have Create External Table permission.                                                                                                                                                           |
| Data source: | Amazon Redshift, Aster, Google BigQuery, Netezza, Snowflake, Yellowbrick                                                                                                                                                                                                                |
| Notes:       | Support for Aster was added in SAS 9.4M2.<br>Support for Amazon Redshift was added in SAS 9.4M6.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Google BigQuery was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">BULKUNLOAD= data set option</a> , <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">Netezza bulk unloading</a> , <a href="#">Aster bulk unloading</a>                                                                                                                  |

---

## Syntax

**BULKUNLOAD=YES | NO**

### Syntax Description

#### **YES**

calls the Remote External Table interface to retrieve data from the server.

#### **NO**

uses standard result sets to retrieve data from the DBMS.

---

## Details

Using BULKUNLOAD=YES is the fastest way to retrieve large numbers of rows from an external table.

*Amazon Redshift*: This value calls the Amazon Redshift UNLOAD command to retrieve data from an Amazon Redshift DBMS.

*Aster*: This value uses the Aster client tool, ncluster\_export, to retrieve data from the Aster DBMS.

*Google BigQuery*: This value calls the Google BigQuery Extractor API.

*Netezza*: This value calls the Netezza Remote External Table interface to retrieve data from the Performance Server.

*Yellowbrick*: This option calls the YBUNLOAD utility.

## CAST= LIBNAME Statement Option

Specifies whether SAS or the Teradata DBMS server performs data conversions.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: none

Data source: Teradata

See: [CAST= data set option](#), [CAST\\_OVERHEAD\\_MAXPERCENT= LIBNAME option](#), [CAST\\_OVERHEAD\\_MAXPERCENT= data set option](#)

## Syntax

**CAST=YES | NO**

### Syntax Description

#### YES

forces data conversions (casting) to be done on the Teradata DBMS server and overrides any data overhead percentage limit.

#### NO

forces data conversions to be done by SAS and overrides any data overhead percentage limit.

## Details

Internally, SAS numbers and dates are floating-point values. Teradata has varying formats for numbers, including integers, floating-point values, and decimal values. Number conversion must occur when you are reading Teradata numbers that are not floating point (Teradata FLOAT). SAS/ACCESS can use the Teradata CAST= function to cause Teradata to perform numeric conversions. The parallelism of Teradata makes it suitable for performing this work. This is especially true when running SAS on z/OS, where CPU activity can be costly.

CAST= can cause more data to be transferred from Teradata to SAS, as a result of the option forcing the Teradata type into a larger SAS type. For example, the CAST= transfer of a Teradata BYTEINT to SAS floating point adds seven overhead bytes to each row transferred.

These Teradata types are candidates for casting.

- INTEGER

- BYTEINT
- SMALLINT
- DECIMAL
- DATE

SAS/ACCESS limits data expansion for CAST= to 20% to trade rapid data conversion by Teradata for extra data transmission. If casting does not exceed a 20% data increase, all candidate columns are cast. If the increase exceeds this limit, SAS attempts to cast only Teradata DECIMAL types. If casting only DECIMAL types still exceeds the increase limit, SAS performs the data conversions.

You can alter the casting rules by using the CAST= or CAST\_OVERHEAD\_MAXPERCENT= LIBNAME option. With CAST\_OVERHEAD\_MAXPERCENT=, you can change the 20% overhead limit. With CAST=, you can override the percentage rules.

- CAST=YES forces Teradata to cast all candidate columns.
- CAST=NO cancels all Teradata casting.

CAST= applies only when you are reading Teradata tables into SAS, not when you are writing Teradata tables from SAS.

Also, CAST= applies only to SQL that SAS generates for you. If you provide your own SQL with the explicit SQL feature of PROC SQL, you must code your own casting clauses. Data conversions are therefore forced to occur in Teradata instead of SAS.

## Examples

### Example 1: Force Casting for All Tables

This example demonstrates the use of the CAST= option in a LIBNAME statement to force casting for all referenced tables.

```
libname mydblib teradata user=myusr1 pw=mypwd1 cast=yes;
proc print data=mydblib.emp;
where empno<1000;
run;
proc print data=mydblib.sal;
where salary>50000;
run;
```

### Example 2: Turn Casting Off for a Specific Table Reference

This example demonstrates the use of the CAST= option in a table reference to turn casting off for that table.

```
proc print data=mydblib.emp (cast=no);
where empno<1000;
run;
```

---

## CAST\_OVERHEAD\_MAXPERCENT= LIBNAME Statement Option

Specifies the overhead limit for data conversions to perform in Teradata instead of SAS.

|              |                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                             |
| Category:    | Data Set Control                                                                                                                         |
| Default:     | 20                                                                                                                                       |
| Data source: | Teradata                                                                                                                                 |
| See:         | <a href="#">CAST= LIBNAME option</a> , <a href="#">CAST= data set option</a> , <a href="#">CAST_OVERHEAD_MAXPERCENT= data set option</a> |

---

## Syntax

**CAST\_OVERHEAD\_MAXPERCENT=*n***

### Syntax Description

*n*

any positive numeric value. The engine default is 20.

---

## Details

Teradata INTEGER, BYTEINT, SMALLINT, and DATE columns require conversion when read in to SAS. Either Teradata or SAS can perform conversions. When Teradata performs the conversion, the row size that is transmitted to SAS using the Teradata CAST operator can increase. **CAST\_OVERHEAD\_MAXPERCENT=** limits the allowable increase, also called *conversion overhead*.

---

## Example: Increase the Allowable Overhead

This example demonstrates the use of **CAST\_OVERHEAD\_MAXPERCENT=** to increase the allowable overhead to 40%.

```
proc print data=mydblib.emp (cast_overhead_maxpercent=40);
  where empno<1000;
  run;
```

---

# CELLPROP= LIBNAME Statement Option

Modifies the metadata and content of a result data set that the MDX command specifies.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: VALUE

Data source: OLE DB

See: ["Accessing OLE DB for OLAP Data"](#)

---

## Syntax

**CELLPROP=FORMATTED\_VALUE | VALUE**

### Syntax Description

#### **FORMATTED\_VALUE**

specifies that the SAS/ACCESS engine returns formatted data values. All columns are specified as CHARACTER.

#### **VALUE**

specifies that the SAS/ACCESS engine tries to return actual data values. If all values in a column are numeric, that column is specified as NUMERIC.

---

## Details

When an MDX command is issued, the resulting data set might have columns that contain one or more types of data values: the actual value of the cell or the formatted value of the cell.

For example, if you issue an MDX command and the resulting data set contains a column named SALARY, the column could contain data values of two types. It could contain numeric values, such as 50000, or it could contain formatted values, such as \$50,000. Specifying the CELLPROP= option determines how the values are specified and the value of the column.

It is possible for a column in a result set to contain both NUMERIC and CHARACTER data values. For example, a data set might return the data values of 50000, 60000, and UNKNOWN. SAS data sets cannot contain both types of data. In this situation, even if you specify CELLPROP=VALUE, the SAS/ACCESS engine specifies the column as CHARACTER and returns formatted values for that column.

## CHAR\_AS\_NCHAR= LIBNAME Statement Option

Specifies the default character type to use for table columns.

|              |                                            |
|--------------|--------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement               |
| Category:    | Data Set Control                           |
| Default:     | NO                                         |
| Data source: | Netezza, OLE DB, SAP HANA                  |
| Note:        | Support for OLE DB was added in SAS 9.4M4. |
| See:         | <a href="#">DBTYPE= data set option</a>    |

## Syntax

**CHAR\_AS\_NCHAR=YES | NO**

### Syntax Description

#### YES

specifies that NCHAR or NVARCHAR be used as the default column type.

#### NO

specifies that CHAR or VARCHAR be used as the default column type.

## Details

Use this option when you cannot use the [DBTYPE= data set option](#) on page 536 for table columns that contain multilingual character data.

## Example: Specify Multilingual Data as the Default

The SAS data set, local\_cust, contains multilingual data in this example.

```
libname net netezza server=mysrv1 database=mydb1 uid=myusr1
      pwd=mypwd1 CHAR_AS_NCHAR=YES;

data net.customers;
  set sas.local_cust;
run;
```

---

## CLIENT\_ENCODING= LIBNAME Statement Option

Specifies the encoding to use when transferring data from your DBMS.

|              |                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                           |
| Category:    | Data Access                                                                                            |
| Default:     | current SAS session encoding                                                                           |
| Restriction: | This option applies only to UNIX environments.                                                         |
| Interaction: | This option overrides any value that you might have set for the PGCLIENTENCODING environment variable. |
| Data source: | PostgreSQL                                                                                             |
| Note:        | Support for this option was added in SAS Viya 3.4.                                                     |
| Examples:    | Set the CLIENT_ENCODING= LIBNAME option:                                                               |

```
libname mydb postgres server=myserver port=5432  
      user=myuser password='mypwd' database=mydb1 client_encoding='utf8';
```

Specify the PGCLIENTENCODING environment variable on PC hosts:

```
PGCLIENTENCODING UTF8
```

Specify the PGCLIENTENCODING environment variable on UNIX hosts:

```
export PGCLIENTENCODING=UTF8
```

Specify the PGCLIENTENCODING environment variable at SAS invocation:

```
sas -set PGCLIENTENCODING=UTF8
```

---

## Syntax

**CLIENT\_ENCODING='encoding'**

### Syntax Description

***encoding***

specifies the encoding to use when transferring data into SAS.

---

## Details

For a list of encoding values, see “[Encodings and Their Aliases and Encoding Character Set Compatibility](#)” in *SAS National Language Support (NLS): Reference Guide*.

---

## CLIENT\_ID= LIBNAME Statement Option

Specifies the client ID value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

|              |                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                          |
| Category:    | Data Access                                                                                                           |
| Default:     | none                                                                                                                  |
| Interaction: | Use this option in conjunction with the CLIENT_SECRET= and REFRESH_TOKEN= options.                                    |
| Data source: | Google BigQuery                                                                                                       |
| Note:        | Support for this option was added in the April 2020 update for SAS/ACCESS.                                            |
| See:         | <a href="#">CLIENT_SECRET= LIBNAME option on page 180</a> , <a href="#">REFRESH_TOKEN= LIBNAME option on page 310</a> |
| Example:     | CLIENT_ID="919191919191-abababa5ab96so8v9o298np4tg0la1md.apps.googleusercontent.com"                                  |

---

## Syntax

**CLIENT\_ID=***ID-value*

### Syntax Description

***ID-value***

specifies the client ID value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

---

## Details

This value is masked in the SAS log.

This option accepts values that have been encoded using PROC PWENCODE. SAS/ACCESS recognizes encoded values as those that begin with a SAS encoding tag. For more information, see “[PWENCODE Procedure](#)” in *Base SAS Procedures Guide*.

---

## CLIENT\_SECRET= LIBNAME Statement Option

Specifies the client secret value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

|           |                              |
|-----------|------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement |
|-----------|------------------------------|

|              |                                                                                                                   |
|--------------|-------------------------------------------------------------------------------------------------------------------|
| Category:    | Data Access                                                                                                       |
| Default:     | none                                                                                                              |
| Interaction: | Use this option in conjunction with the CLIENT_ID= and REFRESH_TOKEN= options.                                    |
| Data source: | Google BigQuery                                                                                                   |
| Note:        | Support for this option was added in the April 2020 update for SAS/ACCESS.                                        |
| See:         | <a href="#">CLIENT_ID= LIBNAME option on page 180</a> , <a href="#">REFRESH_TOKEN= LIBNAME option on page 310</a> |
| Example:     | CLIENT_SECRET=O8XKYBedFZdX_1BAbABab9AB                                                                            |

## Syntax

**CLIENT\_SECRET**=*authentication-secret*

### Syntax Description

#### *authentication-secret*

specifies the client secret value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

## Details

This value is masked in the SAS log.

This option accepts values that have been encoded using PROC PWENCODE. SAS/ACCESS recognizes encoded values as those that begin with a SAS encoding tag. For more information, see “[PWENCODE Procedure](#)” in *Base SAS Procedures Guide*.

## COMMAND\_TIMEOUT= LIBNAME Statement Option

Specifies the number of seconds to wait before a data source command times out.

|              |                                                  |
|--------------|--------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                     |
| Category:    | Data Set Control                                 |
| Default:     | 0                                                |
| Data source: | OLE DB                                           |
| See:         | <a href="#">COMMAND_TIMEOUT= data set option</a> |

## Syntax

**COMMAND\_TIMEOUT=***number-of-seconds*

### Syntax Description

*number-of-seconds*

an integer greater than or equal to 0, where 0 represents no time-out.

---

## CONNECTION= LIBNAME Statement Option

Specifies whether operations on single or multiple librefs can share a connection to the DBMS.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Category:    | Data Access                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Default:     | DBMS-specific                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Interaction: | For DBMSs that default to CONNECTION=UNIQUE, the LIBNAME connection can fail when you use SQL_FUNCTIONS= for that same DBMS to store the external SQL dictionary.                                                                                                                                                                                                                                                                                                          |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                         |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Hadoop was added in SAS 9.4M4.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p>                                                                                         |
| See:         | <a href="#">ACCESS= LIBNAME option</a> , <a href="#">CONNECTION_GROUP= LIBNAME option</a> , <a href="#">DBMSTEMP= LIBNAME option</a> , <a href="#">DEFER= LIBNAME option</a>                                                                                                                                                                                                                                                                                               |
| CAUTION:     | <b>When you use a GLOBAL or single SHARED connection to access multiple tables, performing a commit or rollback on one table that is being updated might affect all other tables that are open for update.</b> Even if you open a table only for READ, its READ cursor might be resynchronized due to this commit or rollback. If the cursor is resynchronized, there is no guarantee that the new solution table matches the original solution table that was being read. |

---

## Syntax

**CONNECTION=**[UNIQUE](#) | [SHAREREAD](#) | [SHARED](#) | [GLOBALREAD](#) | [GLOBAL](#)

## Syntax Description

### **SHAREDREAD**

specifies that all READ operations that access DBMS tables *in a single libref* share a single connection if the conditions for sharing a connection are met. For more information, see “[Conditions for a Shared DBMS Connection](#)”.

A separate connection is established for every table that is opened for update or output operations.

**Tip** Where available, this is usually the default value because it offers the best performance and it guarantees data integrity.

### **UNIQUE**

specifies that a separate connection is established every time a DBMS table is accessed by your SAS application.

**Tip** Use UNIQUE if you want each use of a table to have its own connection.

### **SHARED**

specifies that *all* operations that access DBMS tables *in a single libref* share a single connection if the conditions for sharing a connection are met. For more information, see “[Conditions for a Shared DBMS Connection](#)”.

**Restriction** not valid for MySQL

**Note** The CONNECTION= option controls only connections that you use to open tables with a libref. When you specify CONNECTION=SHARED, it has no effect on utility connections or explicit pass-through connections.

**Tip** Use SHARED to eliminate the deadlock that can occur when you create and load a DBMS table from an existing table that exists in the same database or tablespace. This happens only in certain output processing situations and is the only recommended use for CONNECTION=SHARED.

### **GLOBALREAD**

specifies that all READ operations that access DBMS tables *in multiple librefs* share a single connection if the conditions for sharing a connection are met. For more information, see “[Conditions for a Shared DBMS Connection](#)”.

A separate connection is established for each table that is opened for update or output operations.

### **GLOBAL**

specifies that *all* operations that access DBMS tables *in multiple librefs* share a single connection if the conditions for sharing a connection are met. You must set CONNECTION=GLOBAL to access temporary tables. For more information, see “[Conditions for a Shared DBMS Connection](#)”.

**Restriction** not valid for MySQL

**Interactions** One connection is shared for all tables that any libref references for which you specify CONNECTION=GLOBAL.

To access temporary tables, set CONNECTION=GLOBAL and set the DBMSTEMP= LIBNAME option to YES. (Informix and SAP ASE do not support the DBMSTEMP= LIBNAME option.)

When CONNECTION\_GROUP= is specified, the default value for CONNECTION= is GLOBAL.

|     |                                                                                                                                                                                                               |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tip | When you specify CONNECTION=GLOBAL, any pass-through code that you include after the LIBNAME statement can share the connection. For details, see the <a href="#">CONNECT statement example on page 696</a> . |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Details

### Overview of the CONNECTION= LIBNAME Option

The main reason for using the CONNECTION= LIBNAME option is to control the number of physical connections to your DBMS. When you specify that you want to share DBMS connections, you enable SAS to use one physical connection across multiple DATA steps and procedure calls. In this way, you limit the number of physical connections for your SAS session. Sharing a connection also enables you to share access to temporary tables across DATA steps and procedure calls.

SAS/ACCESS interfaces that support single or multiple simultaneous connections to the DBMS support this option. Not all values are valid for all SAS/ACCESS interfaces.

For most SAS/ACCESS interfaces, there must be a connection, also known as an *attach*, to the DBMS server before a user can access any data. Typically, a DBMS connection has one transaction, or work unit, that is active in the connection. This transaction is affected by any SQL commits or rollbacks that the engine performs within the connection while executing the SAS application.

The CONNECTION= option lets you control the number of connections, and therefore transactions, that your SAS/ACCESS interface executes and supports for each LIBNAME statement, SQL pass-through CONNECT statement, or both.

When you connect to the DBMS server via PROC SQL, only the GLOBAL and SHARED values from a LIBNAME statement are passed down.

## Default Values

*Table 12.1 DBMS-Specific Default Values*

| DBMS                                        | Default      |
|---------------------------------------------|--------------|
| Amazon Redshift                             | SHAREDREREAD |
| DB2 under UNIX and PC Hosts, DB2 under z/OS |              |
| Hadoop                                      |              |
| Impala                                      |              |
| Informix                                    |              |
| JDBC                                        |              |
| MySQL                                       |              |

| DBMS                 | Default                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------|
| OLE DB               |                                                                                                               |
| Oracle               |                                                                                                               |
| PostgreSQL           |                                                                                                               |
| SAP HANA             |                                                                                                               |
| SAP IQ               |                                                                                                               |
| Snowflake            |                                                                                                               |
| Yellowbrick          |                                                                                                               |
| Aster                | UNIQUE                                                                                                        |
| Netezza              |                                                                                                               |
| Vertica              |                                                                                                               |
| Greenplum            | UNIQUE for a data source that supports only one active open cursor per connection; otherwise, SHAREDREAD      |
| HAWQ                 |                                                                                                               |
| Microsoft SQL Server |                                                                                                               |
| ODBC                 |                                                                                                               |
| SAP ASE              |                                                                                                               |
| Teradata             | UNIQUE for network attached systems (UNIX and PC platforms)<br>SHAREDREAD for channel-attached systems (z/OS) |

## Conditions for a Shared DBMS Connection

If you want to share a connection across librefs, the critical connection options that you specify must be the same. You can specify the connection options in a LIBNAME statement or in the CONNECT statement in an SQL procedure call. When SAS/ACCESS compares connection option values, it does not matter whether you use optional quotation marks across libref declarations.

Here are the conditions that must be met to share a physical connection to your DBMS:

- These connection options must have the same value for each libref declaration:

|           |                           |
|-----------|---------------------------|
| USER=     | ACCOUNT=                  |
| PASSWORD= | DATABASE=                 |
| SERVER=   | SCHEMA= (except Teradata) |

Teradata does not require values for SCHEMA= to be the same across libref declarations.

- These LIBNAME options must have the same value for each libref declaration:

|                           |                       |
|---------------------------|-----------------------|
| DATABASE=                 | SQL_FUNCTIONS=        |
| SCHEMA= (except Teradata) | READ_ISOLATION_LEVEL= |

|                   |                         |
|-------------------|-------------------------|
| CONNECTION=       | READ_LOCK_TYPE=         |
| CONNECTION_GROUP= | READ_MODE_WAIT=         |
| DBCONINIT=        | UPDATE_ISOLATION_LEVEL= |
| DBCONTERM=        | UPDATE_LOCK_TYPE=       |
| DBLIBINIT=        | UPDATE_MODE_WAIT=       |
| DBLIBTERM=        |                         |

If any of these conditions are not met, SAS/ACCESS automatically creates additional physical connections to the DBMS.

## Examples

### Example 1: Use SHAREDREAD

In this example, MYDBLIB makes the first connection to the DBMS. This connection is used to print the data from MYDBLIB.TAB. MYDBLIB2 makes the second connection to the DBMS. A third connection is used to update MYDBLIB.TAB. The third connection is closed at the end of the PROC SQL UPDATE statement. The first and second connections are closed with the CLEAR option.

```
/* connection 1 */
libname mydblib oracle user=myusr1 pw=mypwd1
  path='mysrv1' connection=sharedread;
/* connection 2 */
libname mydblib2 oracle user=myusr1
  pw=mypwd1 path='mysrv1' connection=sharedread;
proc print data=mydblib.tab...
/* connection 3 */
proc sql;
  update mydblib.tab...
libname mydblib clear;
libname mydblib2 clear;
```

### Example 2: Use GLOBALREAD

In this example, the two librefs, MYDBLIB and MYDBLIB2, share the same connection for Read access because CONNECTION=GLOBALREAD and the connection options are identical. The first connection prints the data from MYDBLIB.TAB while a second connection updates MYDBLIB.TAB. The second connection is closed at the end of the step. The first connection is closed with the final LIBNAME statement.

```
/* connection 1 */
libname mydblib oracle user=mysrv1 pw=mypwd1
  path='myorapath' connection=globalread;
libname mydblib2 oracle user=mysrv1 pw=mypwd1
  path='myorapath' connection=globalread;
proc print data=mydblib.tab...
/* connection 2 */
proc sql;
  update mydblib.tab...
/* does not close connection 1 */
```

```
libname mydblib clear;
/* closes connection 1 */
libname mydblib2 clear;
```

## Example 3: Use UNIQUE

In this example, the MYDBLIB libref establishes a connection to obtain database information. Another connection is established in order to print the data from MYDBLIB.TAB. That connection is closed at the end of the PRINT procedure. Another connection is established to update MYDBLIB.TAB. That connection is closed at the end of the PROC SQL. The CLEAR option in the LIBNAME statement at the end of this example then closes the connection that was made during the MYDBLIB libref assignment.

```
libname mydblib oracle user=myusr1 pw=mypwd1
  path='mysrv1' connection=unique;
proc print data=mydblib.tab...
proc sql;
  update mydblib.tab...
libname mydblib clear;
```

## Example 4: Use SHARED

In this SHARED example, DB2DATA.NEW is created in the database TEST. The DB2DATA.OLD table exists in the same database. So the CONNECTION=SHARED option lets the DB2 engine share the connection for reading the old table and also creating and loading the new table.

```
libname db2data db2 connection=shared;
data db2data.new (in = 'database test');
  set db2data.old;
run;
```

If you did not use the CONNECTION= option in this case, you would deadlock in DB2 and receive this error.

```
ERROR: Error attempting to CREATE a DBMS table.
ERROR: DB2 execute error DSNT408I SQLCODE = -911,
ERROR: THE CURRENT UNIT OF WORK HAS BEEN ROLLED
      BACK DUE TO DEADLOCK.
```

## Example 5: Use GLOBAL

In this example for DB2, both PROC DATASETS invocations appropriately report "no members in directory." This happens because SESSION.B, as a temporary table, has no entry in the SYSIBM.SYSTABLES system catalog. However, the DATA \_NULL\_ step and SELECT \* from PROC SQL step both return the expected rows. For DB2, when SCHEMA=SESSION, the database first looks for a temporary table before attempting to access any physical schema named SESSION.

```
libname x db2 connection=global schema=SESSION;
proc datasets lib=x;
quit;
/*
*   DBMS-specific code to create a temporary table impervious
*   to commits and populate the table directly in the
```

```

* DBMS from another table.
*/
proc sql;
connect to db2(connection=global schema=SESSION);
execute ( DECLARE GLOBAL TEMPORARY TABLE SESSION.B LIKE SASDXS.A
           ON COMMIT PRESERVE ROWS
         ) by db2;
execute ( insert into SESSION.B select * from SASDXS.A
           ) by db2;
quit;
/* Access the temp table through the global libref. */
data _null_;
set x.b;
put _all_;
run;
/* Access the temp table through the global connection. */
proc sql;
connect to db2 (connection=global schema=SESSION);
select * from connection to db2
( select * from SESSION.B );
quit;
proc datasets lib=x;
quit;

```

In this example, two different librefs share one connection.

```

libname db2lib db2 connection=global;
libname db2data db2 connection=global;
data db2lib.new(in='database test');
      set db2data.old;
run;

```

If you did not use the CONNECTION= option in this last example, you would deadlock in DB2 and receive this error.

```

ERROR: Error attempting to CREATE a DBMS table.
ERROR: DB2 execute error DSNT408I SQLCODE = -911,
ERROR: THE CURRENT UNIT OF WORK HAS BEEN ROLLED
BACK DUE TO DEADLOCK.

```

## CONNECTION\_GROUP= LIBNAME Statement Option

Causes operations on multiple librefs and on multiple SQL pass-through facility CONNECT statements to share a connection to the DBMS.

|              |                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                      |
| Category:    | Data Set Control                                                                                                                                                                                                                                     |
| Default:     | none                                                                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick |

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Notes:   | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                                                    |
| See:     | <a href="#">CONNECTION= LIBNAME option</a>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| CAUTION: | <b>When you use a GLOBAL or single SHARED connection for multiple table opens, performing a commit or rollback on one table that is being updated also applies to all other tables that are open for update.</b> Even if you open a table only for READ, its READ cursor might be resynchronized because of this commit or rollback. If the cursor is resynchronized, the new solution table might not match the original solution table that was being read. |

---

## Syntax

**CONNECTION\_GROUP=***connection-group-name*

### Syntax Description

***connection-group-name***  
name of a connection group.

---

## Details

This option causes a DBMS connection to be shared by all operations on multiple librefs, if all participating librefs that the LIBNAME statements create meet these conditions:

- the same value for the CONNECTION\_GROUP= option
- identical DBMS connection options, as specified in “[Conditions for a Shared DBMS Connection](#)” on page 185

If you specify **CONNECTION=GLOBAL** or **CONNECTION=GLOBALREAD**, operations on multiple librefs can share a connection even if you omit CONNECTION\_GROUP=.

*Informix:* The CONNECTION\_GROUP= option enables multiple librefs or multiple SQL pass-through facility CONNECT statements to share a connection to the DBMS. This overcomes the Release 8.2 limitation where users were unable to access scratch tables across step boundaries as a result of new connections being established with every procedure.

## Example: Share a Connection with Identical Connection Options

In this example, the MYDBLIB libref shares a connection with MYDBLIB2 by specifying CONNECTION\_GROUP=MYGROUP and by specifying identical connection options. The libref MYDBLIB3 makes a second connection to another connection group called ABC. The first connection is used to print the data from and also for updating MYDBLIB.TAB. The third connection is closed at the end of the step. The first connection is closed by the final LIBNAME statement for that connection. Similarly, the second connection is closed by the final LIBNAME statement for that connection.

```
/* connection 1 */
libname mydblib oracle user=myusr1
    pw=mypwd1
    connection_group=mygroup;
libname mydblib2 oracle user=myusr1
    pw=mypwd1
    connection_group=mygroup;
/* connection 2 */
libname mydblib3 oracle user=myusr1
    pw=mypwd1
    connection_group=abc;
proc print data=mydblib.tab...
/* connection 1 */
proc sql;
    update mydblib.tab...
/* does not close connection 1*/
libname mydblib clear;
/* closes connection 1 */
libname mydblib2 clear;
/* closes connection 2 */
libname mydblib3 clear;
```

---

## CONOPTS= LIBNAME Statement Option

Specifies database-specific connection options to use when connecting.

|              |                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                              |
| Category:    | Data Set Control                                                                                                             |
| Default:     | none                                                                                                                         |
| Data source: | Amazon Redshift, Aster, Greenplum, Impala, Netezza, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick            |
| Notes:       | Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| Tip:         | You can also specify any of these additional connection options: DATABASE=, ROLE=, SCHEMA=, and WAREHOUSE=. [Snowflake]      |
| Example:     | This example specifies an SSL mode of <i>required</i> for PostgreSQL.                                                        |

```
libname pg postgres user=myusrl pwd=mypwd1 server='mysrv.com'
port=5432 db=userpg conopts='sslmode=required';
```

---

## Syntax

**CONOPTS='connection\_options';**

---

# CURSOR\_TYPE= LIBNAME Statement Option

Specifies the cursor type for read-only and updatable cursors.

|              |                                                                                                                                                                                    |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement and some DBMS-specific connection options. See the DBMS-specific reference section for details.                                                       |
| Category:    | Data Set Control                                                                                                                                                                   |
| Alias:       | CURSOR= [Impala, SAP IQ]                                                                                                                                                           |
| Default:     | DBMS- and operation-specific                                                                                                                                                       |
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Microsoft SQL Server, ODBC, OLE DB, PostgreSQL, SAP IQ                                                      |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift and PostgreSQL was added in SAS Viya 3.4.                                                                  |
| See:         | <a href="#">COMMAND_TIMEOUT= LIBNAME option</a> , <a href="#">CURSOR_TYPE= data set option</a> , <a href="#">KEYSET_SIZE= data set option</a> [only Microsoft SQL Server and ODBC] |

---

## Syntax

**CURSOR\_TYPE=DYNAMIC | FORWARD\_ONLY | KEYSET\_DRIVEN | STATIC**

### Syntax Description

#### DYNAMIC

specifies that the cursor reflects all changes that are made to the rows in a result set as you move the cursor. The data values and the membership of rows in the cursor can change dynamically on each fetch.

**Note:** This value is not valid for Impala.

---

#### FORWARD\_ONLY

specifies that the cursor functions like a DYNAMIC cursor except that it supports only sequential fetching of rows.

**Note:** This value is not valid for OLE DB.

**KEYSET\_DRIVEN**

specifies that the cursor determines which rows belong to the result set when the cursor is opened. However, changes that are made to these rows are reflected as you scroll around the cursor.

---

**Note:** This value is not valid for Impala.

---

**STATIC**

specifies that the cursor builds the complete result set when the cursor is opened. No changes that are made to the rows in the result set after the cursor is opened are reflected in the cursor. Static cursors are read-only.

---

## Details

Not all drivers support all cursor types. An error is returned if the specified cursor type is not supported. The driver is allowed to modify the default without an error. See your database documentation for more information.

When no options have been specified yet, here are the initial DBMS-specific defaults:

**Table 12.2** DBMS-Specific Defaults for CURSOR\_TYPE=

|                             | Greenplum<br>HAWQ<br>Impala | Amazon Redshift<br>Microsoft SQL Server<br>PostgreSQL<br>SAP IQ |
|-----------------------------|-----------------------------|-----------------------------------------------------------------|
| DB2 under UNIX and PC Hosts | OLE DB                      |                                                                 |
| none                        | FORWARD_ONLY                | DYNAMIC                                                         |

Here are the operation-specific defaults:

**Table 12.3** Operation-Specific Defaults for CURSOR\_TYPE=

| Operation                    | DB2 under UNIX and PC Hosts | Impala, ODBC, SAP IQ | Microsoft SQL Server             | OLE DB       | Greenplum, HAWQ |
|------------------------------|-----------------------------|----------------------|----------------------------------|--------------|-----------------|
| insert<br>(UPDATE_SQL=NO)    | KEYSET_DRIVEN               | KEYSET_DRIVEN        | DYNAMIC                          | FORWARD_ONLY | FORWARD_ONLY    |
| read<br>(such as PROC PRINT) | driver default              |                      | driver default<br>(FORWARD_ONLY) |              | FORWARD_ONLY    |
| update<br>(UPDATE_SQL=NO)    | KEYSET_DRIVEN               | KEYSET_DRIVEN        | DYNAMIC                          | FORWARD_ONLY | FORWARD_ONLY    |

| Operation             | DB2 under UNIX and PC Hosts | Impala, ODBC, SAP IQ | Microsoft SQL Server | OLE DB  | Greenplum, HAWQ |
|-----------------------|-----------------------------|----------------------|----------------------|---------|-----------------|
| CONNECTION=GL<br>OBAL | none                        | none                 | DYNAMIC              | DYNAMIC | FORWARD_ONLY    |
| CONNECTION=SH<br>ARED |                             |                      |                      |         |                 |

**OLE DB:** Here are the OLE DB properties that are applied to an open rowset. For more information, see your OLE DB programmer reference documentation:

**Table 12.4 OLE DB Properties for CURSOR\_TYPE= Values**

| CURSOR_TYPE=                               | OLE DB Properties Applied                                   |
|--------------------------------------------|-------------------------------------------------------------|
| FORWARD_ONLY or DYNAMIC<br>(see “Details”) | DBPROP_OTHERINSERT=TRUE,<br>DBPROP_OTHERUPDATEDELETE=TRUE   |
| KEYSET_DRIVEN                              | DBPROP_OTHERINSERT=FALSE,<br>DBPROP_OTHERUPDATEDELET=TRUE   |
| STATIC                                     | DBPROP_OTHERINSERT=FALSE,<br>DBPROP_OTHERUPDATEDELETE=FALSE |

## DATETIME2= LIBNAME Statement Option

Specifies the scale for the timestamp literal for Microsoft SQL Server 2008 and the native Microsoft driver.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: NO

Data source: Microsoft SQL Server, ODBC

Notes: Support for this LIBNAME option was added for SAS 9.4.

Support for Microsoft SQL Server was added in SAS Viya 3.4.

See: [DATETIME2= data set option](#)

## Syntax

**DATETIME2=YES | NO**

## Syntax Description

### YES

specifies a DATETIME precision and scale when creating the timestamp for the WHERE clause.

### NO

uses the first occurring DATETIME precision and scale when creating the timestamp for the WHERE.

## Details

When this value is not specified, SAS/ACCESS generates timestamp values in WHERE clauses in a format that matches the format of the column that is being used to subset data.

## DB\_LENGTH\_SEMANTICS\_BYTEx LIBNAME Statement Option

Indicates whether CHAR and VARCHAR2 column lengths are specified in bytes or characters when creating an Oracle table.

|              |                                                    |
|--------------|----------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                       |
| Category:    | Data Set Control                                   |
| Default:     | YES                                                |
| Supports:    | NLS                                                |
| Data source: | Oracle                                             |
| See:         | <a href="#">SAS/ACCESS LIBNAME options for NLS</a> |

## Syntax

**DB\_LENGTH\_SEMANTICS\_BYTEx YES | NO**

## Syntax Description

### YES

specifies that CHAR and VARCHAR2 column lengths are specified in bytes when creating an Oracle table. The byte length is derived by multiplying the number of characters in SAS with DBSERVER\_MAX\_BYTEx value.

### NO

specifies that CHAR and VARCHAR2 column lengths are specified in characters when creating an Oracle table. The CHAR keyword is also added next to the length value to indicate that this is the character (not byte) length. For fixed-width encoding, the number of characters is derived by dividing the byte length in SAS

for the variable by the value in **DBCLIENT\_MAX\_BYTES=**. For variable-width encoding, the number of characters remains the same as the number of bytes.

---

## Details

This option is appropriate only when creating Oracle tables from SAS. It is therefore ignored in other contexts, such as reading or updating tables.

Length values chosen for variable-width encodings might be more than what is actually needed.

---

## DB\_OBJECTS= LIBNAME Statement Option

Specifies which database objects to return with PROC DATASETS or in SAS Explorer.

|              |                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                 |
| Category:    | Data Set Control                                                                                                                             |
| Default:     | (TABLES VIEWS) This is set dynamically.                                                                                                      |
| Restriction: | Because SHOW_SYNONYMS=YES overrides DB_OBJECTS= and is available only for backward compatibility, you should instead use DB_OBJECT=SYNONYMS. |
| Data source: | Oracle                                                                                                                                       |
| See:         | <a href="#">SHOW_SYNONYMS= LIBNAME option</a>                                                                                                |

---

## Syntax

**DB\_OBJECTS=TABLES | VIEWS | SYNONYMS | PUBLIC\_SYNONYMS | ALL**

### Syntax Description

#### **ALL**

returns all database object names, which can slow performance. Specify ALL by itself. It always overrides any multiple values that you specify.

#### **PUBLIC\_SYNONYMS**

returns only public database synonym names.

#### **SYNONYMS**

returns only database synonym names.

#### **TABLES**

returns only database table names.

#### **VIEWS**

returns only database view names.

## Example: Specify Multiple Objects

```
DB_OBJECTS= (VIEWS SYNONYMS)
```

## DBCLIENT\_ENCODING\_FIXED= LIBNAME Statement Option

Specifies whether SAS session encoding is a fixed width.

|              |                                                                              |
|--------------|------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                 |
| Category:    | Data Set Control                                                             |
| Default:     | YES or NO, based on SAS encoding                                             |
| Restriction: | Applies only when you create Oracle tables from within SAS                   |
| Supports:    | NLS                                                                          |
| Data source: | Oracle                                                                       |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M1.                      |
| See:         | <a href="#">SAS/ACCESS LIBNAME options for NLS</a>                           |
| CAUTION:     | <b>To avoid truncation issues, use care when setting this option to YES.</b> |

## Syntax

**DBCLIENT\_ENCODING\_FIXED=YES | NO**

### Syntax Description

#### YES

indicates that database table column lengths in characters are a fixed width. Use this value to adjust byte lengths within SAS for any database column lengths that are specified in bytes. The number of characters is calculated as the length that is specified in SAS, divided by the value in [DBCLIENT\\_MAX\\_BYTES=](#). If needed, and if [DB\\_LENGTH\\_SEMANTIC\\_BYTE=YES](#), the character length is then multiplied by the value in [DBSERVER\\_MAX\\_BYTES=](#).

#### NO

indicates that database table column lengths are not a fixed width.

## Example: Specify SAS Session Encoding

For this example, the SAS session encoding is euc-cn, the locale is Chinese, and the Oracle server encoding is UTF8.

```
libname lib1 oracle path=mypath1 user=myusr1 pw=mypwd1
```

```
      dbserver_max_bytes=3 dbclient_max_bytes=2
      dbclient_encoding_fixed=yes;
```

```
      data lib1.test;
      id='中文';
      run;
```

```
SQL> desc test;
```

| Name | Null? | Type     |
|------|-------|----------|
| ID   |       | CHAR (6) |

```
libname lib2 oracle path=nlsbip08 user=myusr1 pw=mypwd1
      dbserver_max_bytes=3 dbclient_max_bytes=2
      dbclient_encoding_fixed=no;
```

```
SQL> desc test;
```

| Name | Null? | Type      |
|------|-------|-----------|
| ID   |       | CHAR (12) |

## DBCLIENT\_MAX\_BYTES= LIBNAME Statement Option

Specifies the maximum number of bytes per single character in the database client encoding, which usually matches SAS encoding.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Alias:       | CLIENT_MAX_BYTES= [Impala]                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Defaults:    | matches the maximum bytes per single character of SAS session encoding [Amazon Redshift, Aster, Google BigQuery, Impala, MySQL, Oracle, PostgreSQL, Snowflake, Vertica, Yellowbrick]<br>1 [DB2 under UNIX and PC Hosts, SAP IQ, Teradata]                                                                                                                                                                                                        |
| Supports:    | NLS                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Impala, MySQL, Oracle, PostgreSQL, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                                                                                                       |
| Notes:       | Support for Google BigQuery was added in the April 2020 update of SAS/ACCESS for SAS 9.4M6 and SAS Viya 3.5.<br>Support for MySQL, Teradata, and Vertica was added for SAS 9.4.<br>Support for PostgreSQL was added in SAS 9.4M1.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for MySQL was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS. |

Support for Yellowbrick was added in SAS 9.4M7.

Support for SAP IQ was added in SAS 9.4M8.

See: [SAS/ACCESS LIBNAME options for NLS](#)

Example: [DBSERVER\\_MAX\\_BYTES= LIBNAME option](#)

## Syntax

**DBCLIENT\_MAX\_BYTES**=*maximum-client-bytes*

### Required Argument

#### ***maximum-client-bytes***

specifies the multiplying factor to apply for storage of CHAR and NCHAR variables for client encoding.

## Details

When using a multi-byte SAS session encoding, such as UTF8 or EUC-CN, it might be necessary to control the number of bytes that SAS allocates for each character in a double-byte or multi-byte character set. By default, SAS estimates how many bytes per character are required to avoid truncation or transcoding issues. This estimation might lead to overallocation.

For example, if a DBMS system contains single-byte ASCII data, but a column's character set encoding is UTF8, SAS overallocates space because it assumes that the UTF8 source requires multiple bytes per character. This overallocation can cause an inflation issue when copying data back and forth between SAS data sets and the external DBMS. It can also lead to performance issues because of the additional blank padding that SAS must manage.

To address this situation, set DBCLIENT\_MAX\_BYTEn=1. This tells SAS that the underlying data is encoded in a single-byte encoding.

The default session encoding for SAS Viya is UTF8. However, if SAS is running in a single-byte session encoding, then this option has no effect because SAS is limited to one byte per character by the encoding.

## Example: Specify a Value for DBCLIENT\_MAX\_BYTEn=

This example shows how to use macro variables in your LIBNAME statement.

```
%let engine=oracle;
%let connopt=path=myPath user=myUsr1 pw=myPwd1;

libname x5 &engine &connopt
      dbclient_max_bytes=3 dbclient_encoding_fixed=yes;
```

```
proc contents data=x5.char_sem; run;
proc contents data=x5.nchar_sem; run;
proc contents data=x5.byte_sem; run;
proc contents data=x5.mixed_sem; run;
```

## DBCOMMIT= LIBNAME Statement Option

Causes an automatic COMMIT (permanently writing data to the DBMS) after processing a specified number of rows.

|               |                                                                                                                                                                                                                                                                                                                         |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                            |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                        |
| Aliases:      | DBINITCMD=, INITCMD= [Oracle]<br>CHECKPOINT= [Teradata]                                                                                                                                                                                                                                                                 |
| Defaults:     | 1000 when a table is created and rows are inserted in a single step (DATA step)<br>0 when rows are inserted, updated, or deleted from an existing table ( PROC APPEND or PROC SQL inserts, updates, or deletions)<br>none [Snowflake]                                                                                   |
| Restriction:  | When you specify a value for DBCOMMIT=, SAS/ACCESS fails for any update that uses a WHERE clause.                                                                                                                                                                                                                       |
| Interactions: | This option is not honored when the DBIDIRECTEXEC system option is enabled.<br>When both options are specified, DBCOMMIT= overrides ERRLIMIT=. See Details.                                                                                                                                                             |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                                           |
| Notes:        | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Impala was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.             |
| See:          | <a href="#">BULKLOAD= data set option</a> , <a href="#">CONNECTION= LIBNAME option</a> , <a href="#">DBCOMMIT= data set option</a> , <a href="#">ERRLIMIT= data set option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">ML_CHECKPOINT= data set option</a> , <a href="#">Using FastLoad</a> |

## Syntax

**DBCOMMIT=*n***

### Syntax Description

*n*

specifies the number of rows that are processed. This value must be an integer greater than or equal to 0.

## Details

**DBCOMMIT=** affects update, delete, and insert processing. The number of rows that are processed includes rows that are not processed successfully. Usually, when you specify DBCOMMIT=0, COMMIT is issued only once: after a procedure or DATA step completes. However, the commit is performed after each statement when you use the SQL procedure.

---

**Note:** If you specify both DBCOMMIT= and ERRLIMIT=, and these options collide during processing, then COMMIT is issued first and ROLLBACK is issued second. Because COMMIT is issued (through the DBCOMMIT= option) before ROLLBACK (through the ERRLIMIT= option), DBCOMMIT= overrides ERRLIMIT=.

---

**DB2 under UNIX and PC Hosts:** When BULKLOAD=YES, the default is 10000.

**Teradata:** For more information, see “[FastLoad Supported Features and Restrictions](#)”. DBCOMMIT= and ERRLIMIT= are disabled for MultiLoad to prevent any conflict with the ML\_CHECKPOINT= data set option.

**Vertica:** For updates or deletions, any value for DBCOMMIT= is reset to 0, because SAS/ACCESS can commit changes only at the end of an action. Also, for updates or deletions, if a value is not already specified for the CONNECTION= LIBNAME option, then CONNECTION= is set to UNIQUE.

---

## DBCONINIT= LIBNAME Statement Option

Specifies a user-defined initialization command to execute immediately after every connection to the DBMS that is within the scope of the LIBNAME statement or libref.

|              |                                                                                                                                                                                                                                                                                                                                                                   |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                      |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                  |
| Aliases:     | DBINITCMD [Vertica]<br>INITCMD [Vertica]                                                                                                                                                                                                                                                                                                                          |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                              |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                         |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Hadoop and JDBC was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Spark was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">AUTHID= LIBNAME option</a> , <a href="#">AUTHID= data set option</a> , <a href="#">DBCONTERM= LIBNAME option</a> , <a href="#">SCHEMA= LIBNAME option</a> , <a href="#">SCHEMA= data set option</a>                                                                                                                                                   |

## Syntax

**DBCONINIT=<'>*DBMS-user-command*<'>**

### Syntax Description

#### ***DBMS-user-command***

any valid command that the SAS/ACCESS engine can execute and that does not return a result set or output parameters.

---

## Details

The initialization command that you select can be a stored procedure or any DBMS SQL statement that might provide additional control over the interaction between your SAS/ACCESS interface and the DBMS.

The command executes immediately after each DBMS connection is successfully established. If the command fails, a disconnection occurs and the libref is not assigned. You must specify the command as a single quoted string.

---

**Note:** The initialization command might execute more than once because one LIBNAME statement might have multiple connections (for example, one for reading and one for updating).

---

**DB2 under z/OS:** If you specify `DBCONINIT="SET CURRENT SQLID='user-ID'"`, then any values that are specified for AUTHID= or SCHEMA= are ignored. Only the value that you specify for SQLID= is used in SQL or procedures that are passed to the database.

---

## Examples

### Example 1: Apply the SET Statement to Every Connection

In this example, the DBCONINIT= option causes the DBMS to apply the SET statement to every connection that uses the MYDBLIB libref.

```
libname mydblib db2
      dbconinit="SET CURRENT SQLID='myauthid'";
proc sql;
  select * from mydblib.customers;
  insert into mydblib.customers
    values('33129804', 'VA', '22809', 'USA',
          '540/545-1400', 'BENNETT SUPPLIES', 'M. JONES',
          '2199 LAUREL ST', 'ELKTON', '22APR97'd);
  update mydblib.invoices
    set amtbill = amtbill*1.10
```

```

      where country = 'USA';
      quit;

```

## Example 2: Pass a Stored Procedure

In this example, a stored procedure is passed to DBCONINIT=.

```

libname mydblib oracle user=myusr1 pass=mypwd1
      dbconinit="begin dept_test(1001,25) ";
end;

```

The SAS/ACCESS engine retrieves the stored procedure and executes it.

## Example 3: Treat Backslash Characters as Literals

In this example, specify that a backslash character ('\\') should be read as a literal character rather than as an escape character. By default, the DBMS variable that controls how the backslash is read is disabled, resulting in a backslash being treated as an escape character. If this is not the desired behavior (such as when specifying a directory path), you can change the behavior.

The command that you specify varies based on your DBMS. For Greenplum, specify the following command for DBCONINIT=:

```
dbconinit="SET standard_conforming_strings = 'ON'"
```

For Aster, specify this command for DBCONINIT=:

```
dbconinit="set session enable_backslash_escapes='off'"
```

For MySQL, specify this command for DBCONINIT=:

```
dbconinit="set session sql_mode='no_backslash_escapes'"
```

## DBCONTERM= LIBNAME Statement Option

Specifies a user-defined termination command to execute before every disconnect from the DBMS that is within the scope of the LIBNAME statement or libref.

|              |                                                                                                                                                                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                                           |
| Category:    | Data Set Control                                                                                                                                                                                                                                                          |
| Aliases:     | DBTERMCMD [Vertica]<br>TERMCMD [Vertica]                                                                                                                                                                                                                                  |
| Default:     | none                                                                                                                                                                                                                                                                      |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Hadoop and JDBC was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.         |

Support for Spark was added in SAS 9.4M7.

Support for Yellowbrick was added in SAS 9.4M7.

See: [DBCONINIT= LIBNAME option](#)

---

## Syntax

**DBCONTERM=<'>*DBMS-user-command*<'>**

### Syntax Description

#### ***DBMS-user-command***

any valid command that the SAS/ACCESS engine can execute and that does not return a result set or output parameters.

---

## Details

The termination command that you select can be a stored procedure or any DBMS SQL statement that might provide additional control over the interaction between the SAS/ACCESS engine and the DBMS. The command executes immediately before SAS terminates each connection to the DBMS. If the command fails, SAS provides a warning message, but SAS still unassigns the library and disconnects from the DBMS. You must specify the command as a single quoted string.

---

**Note:** The termination command might execute more than once because one LIBNAME statement might have multiple connections (for example, one for reading and one for updating).

---

---

## Examples

### Example 1: Drop a Table Before Disconnecting

In this example, the DBMS drops the Q1\_SALES table before SAS disconnects from the DBMS.

```
libname mydblib db2 user=myusr1 using=mypwd1  
      datasrc=invoice dbconterm='drop table q1_sales';
```

### Example 2: Execute a Stored Procedure at Each DBMS Connection

In this example, the stored procedure, SALESTAB\_STORED\_PROC, is executed each time SAS connects to the DBMS, and the BONUSES table is dropped when SAS terminates each connection.

```
libname mydblib db2 user=myusrl
      using=mypwd1 datasrc=sales
      dbconinit='exec salestab_stored_proc'
      dbconterm='drop table bonuses';
```

## DBCREATE\_TABLE\_EXTERNAL= LIBNAME Statement Option

Specifies whether a Hive table is created and stored in the Hive warehouse or external to the Hive warehouse.

|              |                                                                                                                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                 |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                             |
| Aliases:     | DBCREATE_EXTERNAL=                                                                                                                                                                                                                                                                                           |
|              | DBCREATE_EXT=                                                                                                                                                                                                                                                                                                |
| Default:     | NO                                                                                                                                                                                                                                                                                                           |
| Interaction: | This option should be specified with the DBCREATE_TABLE_LOCATION= LIBNAME option.                                                                                                                                                                                                                            |
| Data source: | Hadoop                                                                                                                                                                                                                                                                                                       |
| See:         | <a href="#">DBCREATE_TABLE_LOCATION= LIBNAME option on page 205</a> ,<br><a href="#">DBCREATE_TABLE_OPTS= LIBNAME option</a> , <a href="#">DBCREATE_TABLE_EXTERNAL= data set option</a> , <a href="#">DBCREATE_TABLE_LOCATION= data set option</a> ,<br><a href="#">DBCREATE_TABLE_OPTS= data set option</a> |

## Syntax

**DBCREATE\_TABLE\_EXTERNAL=YES | NO**

### Syntax Description

#### YES

creates an *external* table—one that is stored outside of the Hive warehouse.

#### NO

creates a *managed* table—one that is managed within the Hive warehouse.

## Details

Specifying DBCREATE\_TABLE\_EXTERNAL=YES causes SAS to include the EXTERNAL keyword before the table name when creating a Hive table.

## Example: Create a Hive Table Outside of the Hive Warehouse

In this example, the combination of DBCREATE\_TABLE\_LOCATION='/tmp/extdir' and DBCREATE\_TABLE\_EXTERNAL=YES alters the CREATE TABLE query generated by the SAS DATA step.

```
libname x HADOOP user=xxxxxx pwd=xxxxxx
driverclass=com.cloudera.hive.jdbc.HS2Driver
DBCREATE_TABLE_EXTERNAL=yes
dbcREATE_table_location='/tmp/extdir'
url='jdbc:hive2://host:port/
database;useNativeQuery=1;defaultStringColumnLength=2048;
hive.exec.drop.ignorenonexistent=false';

data x.class;
  set sashelp.class;
run;
```

Here is an example of the log

```
HADOOP_6: Executed: on connection 2
CREATE EXTERNAL TABLE `default`.`CLASS` (`Name` VARCHAR(8), `Sex` VARCHAR(1), `Age` DOUBLE, `Height` DOUBLE, `Weight` DOUBLE)
LOCATION '/tmp/extdir/class' TBLPROPERTIES ('SAS OS Name'='Linux', 'SAS Version'='V.04.00M0D03312022')
```

---

## DBCREATE\_TABLE\_LOCATION= LIBNAME Statement Option

Specifies the default location for external tables.

|              |                                                                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                       |
| Category:    | Data Set Control                                                                                                                                                                                                   |
| Aliases:     | DBCREATE_LOCATION=<br>DBCREATE_LOC=<br>DBCREATE_PATH=                                                                                                                                                              |
| Default:     | None                                                                                                                                                                                                               |
| Requirement: | You must specify the DBCREATE_TABLE_EXTERNAL= LIBNAME option when you specify this option.                                                                                                                         |
| Data source: | Hadoop                                                                                                                                                                                                             |
| Note:        | Support for this option starts in SAS 9.4M8.                                                                                                                                                                       |
| See:         | <a href="#">DBCREATE_TABLE_EXTERNAL= LIBNAME option on page 204</a> ,<br><a href="#">DBCREATE_TABLE_OPTS= LIBNAME option on page 207</a> ,<br><a href="#">DBCREATE_TABLE_LOCATION= data set option on page 508</a> |

## Syntax

**DBCREATE\_TABLE\_LOCATION='path'**

### Syntax Description

**'path'**

specifies the default location for storing external tables.

### Details

This option causes SAS to include the LOCATION keyword in its CREATE TABLE query, along with the specified path. When using the LIBNAME option and creating a new table, the table's name is appended to the user's path.

---

**Note:** The usage of the DBCREATE\_TABLE\_LOCATION= LIBNAME option is slightly different than the DBCREATE\_TABLE\_LOCATION= data set option. The LIBNAME option specifies the default location for any CREATE TABLE queries generated by SAS. SAS appends the table's name to the path, resulting in a fully qualified path. When using the data set option, you must specify the absolute path, including the table's name.

---

### Example: Create a Hive Table outside of the Hive Warehouse

In this example, the combination of DBCREATE\_TABLE\_LOCATION='/tmp/extdir' and DBCREATE\_TABLE\_EXTERNAL=YES alters the CREATE TABLE query generated by the SAS DATA step.

```
libname x HADOOP user=xxxxxx pwd=xxxxxx
driverclass=com.cloudera.hive.jdbc.HS2Driver
DBCREATE_TABLE_EXTERNAL=yes
dbcreate_table_location='/tmp/extdir'
url='jdbc:hive2://host:port/
database;useNativeQuery=1;defaultStringColumnLength=2048;
hive.exec.drop.ignorenonexistent=false';

data x.class;
  set sashelp.class;
run;
```

```
HADOOP_6: Executed: on connection 2
CREATE EXTERNAL TABLE `default`.`CLASS` (`Name` VARCHAR(8), `Sex` VARCHAR(1), `Age` DOUBLE, `Height` DOUBLE, `Weight` DOUBLE)
LOCATION '/tmp/extdir/class' TBLPROPERTIES ('SAS OS Name'='Linux', 'SAS
Version'='V.04.00MOD03312022')
```

---

# DBCREATE\_TABLE\_OPTS= LIBNAME Statement Option

Specifies DBMS-specific syntax to add to the CREATE TABLE statement.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME and CONNECT statements                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Alias:       | POST_STMT_OPTS=                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                                                                                                                             |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Spark was added in SAS 9.4M7.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p>                                                                                  |
| Tips:        | <p>If you are already using DBTYPE= within an SQL CREATE TABLE statement, you can also use it to include column modifiers.</p> <p>If you want all output tables to be in the default (non-TEXTFILE) format, see the examples in this section.</p>                                                                                                                                                                                                                                      |
| See:         | <a href="#">DBCREATE_TABLE_EXTERNAL= LIBNAME option</a> , <a href="#">DBCREATE_TABLE_EXTERNAL= data set option</a> , <a href="#">DBCREATE_TABLE_LOCATION= data set option</a> , <a href="#">DBCREATE_TABLE_OPTS= data set option</a> , <a href="#">DBTYPE= data set option</a> , <a href="#">POST_STMT_OPTS= data set option</a> , <a href="#">POST_TABLE_OPTS= data set option</a> , <a href="#">PRE_STMT_OPTS= data set option</a> , <a href="#">PRE_TABLE_OPTS= data set option</a> |

---

## Syntax

**DBCREATE\_TABLE\_OPTS='DBMS-SQL-clauses'**

### Required Argument

#### **DBMS-SQL-clauses**

specifies one or more DBMS-specific clauses that can be appended to the end of an SQL CREATE TABLE statement.

## Details

You can use this option to add DBMS-specific clauses to the end of the SQL CREATE TABLE statement. The SAS/ACCESS engine passes the SQL CREATE TABLE statement and its clauses to the DBMS. The DBMS then executes the statement and creates the DBMS table. This option applies only when you are creating a DBMS table by specifying a libref that is associated with DBMS data.

If you need to add an option in a location other than at the end of your CREATE TABLE statement, use one of these data set options: POST\_TABLE\_OPTS=, PRE\_STMT\_OPTS=, and PRE\_TABLE\_OPTS=. For example, for Greenplum, a WITH clause should appear after the table name but before a DISTRIBUTED RANDOMLY clause in a CREATE TABLE statement. You should therefore specify a WITH clause using the POST\_TABLE\_OPTS= data set option.

## Examples

### Example 1: Create All Hive Tables in ORC Format

```
libname x hadoop ... DBCREATE_TABLE_OPTS="stored as ORC";
```

### Example 2: Create All Hive Tables in RCFILE Format

```
libname x hadoop ... DBCREATE_TABLE_OPTS="stored as RCFILE";
```

### Example 3: Create All Hive Tables in SEQUENCEFILE Format

```
libname x hadoop ... DBCREATE_TABLE_OPTS="stored as SEQUENCEFILE";
```

---

## DBGEN\_NAME= LIBNAME Statement Option

Specifies how SAS automatically renames to valid SAS variable names any DBMS columns that contain characters that SAS does not allow.

|              |                                                                                                                                                                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                                                     |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                    |
| Default:     | DBMS                                                                                                                                                                                                                                                                                |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.                                                                                                                                                                                                                                            |

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for JDBC was added in SAS Viya 3.4.

Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

See:

[DBGEN\\_NAME= data set option](#), [VALIDVARNAME= system option](#)

---

## Syntax

**DBGEN\_NAME=DBMS | SAS**

### Syntax Description

#### **DBMS**

specifies that SAS renames DBMS columns to valid SAS variable names. SAS converts to underscores any characters that it does not allow. If it converts a column to a name that already exists, it appends a sequence number at the end of the new name.

#### **SAS**

specifies that SAS converts DBMS columns that contain characters that SAS does not allow into valid SAS variable names. SAS uses the format `_COLn`, where *n* is the column number, starting with 0. If SAS converts a name to a name that already exists, it appends a sequence number at the end of the new name.

---

## Details

SAS retains column names when it reads data from DBMS tables unless a column name contains characters that SAS does not allow, such as `$` or `@`. SAS allows alphanumeric characters and the underscore (`_`).

This option is intended primarily for National Language Support, notably for the conversion of kanji to English characters. English characters that are converted from kanji are often those that SAS does not allow. Although this option works for the single-byte character set (SBCS) version of SAS, SAS ignores it in the double-byte character set (DBCS) version. So if you have the DBCS version, you must first specify `VALIDVARNAME=ANY` before using your language characters as column variables.

---

## Example

If you specify `DBGEN_NAME=SAS`, SAS renames a DBMS column named `Dept$Amt` to `_COLn`. If you specify `DBGEN_NAME=DBMS`, SAS renames the `Dept$Amt` column to `Dept_Amt`.

---

## DBINDEX= LIBNAME Statement Option

Improves performance when processing a join that involves a large DBMS table and a small SAS data set.

|              |                                                                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                             |
| Category:    | Data Set Control                                                                                                                                                                                         |
| Default:     | NO                                                                                                                                                                                                       |
| Restriction: | <i>Oracle</i> : Use this option only when the object is a TABLE, not a VIEW. Use DBKEY= when you do not know whether the object is a TABLE.                                                              |
| Data source: | Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica, Yellowbrick |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                              |
| See:         | <a href="#">DBINDEX= data set option</a> , “ <a href="#">MULTI_DATASRC_OPT=</a> ”                                                                                                                        |

---

## Syntax

**DBINDEX=YES | NO**

### Syntax Description

#### YES

specifies that SAS uses columns that have specified DBMS indexes in the WHERE clause.

#### NO

specifies that SAS does not use indexes that are specified on DBMS columns.

---

## Details

When you process a join that involves a large DBMS table and a relatively small SAS data set, you might be able to use DBINDEX= to improve performance.

---

#### CAUTION

Improper use of this option can degrade performance.

---

## Example: Use DBINDEX= in a LIBNAME Statement

```
libname mydblib oracle user=myuser password=userpwd dbindex=yes;
proc sql;
select * from s1 aa, mydblib.dbtab bb where aa.a=bb.a;
select * from s1 aa, mydblib.dbtab bb where aa.a=bb.a;
```

The DBINDEX= values for table Dbtab are retrieved from the DBMS and compared with the join values. In this case, a match was found so that the join is passed down to the DBMS using the index. If the index a was not found, the join would take place in SAS.

---

## DBLIBINIT= LIBNAME Statement Option

Specifies a user-defined initialization command to execute once within the scope of the LIBNAME statement or libref that established the first connection to the DBMS.

|              |                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                 |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                             |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                                                         |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                                                    |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Hadoop and JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Spark was added in SAS 9.4M7.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">CONNECTION= LIBNAME option</a> , <a href="#">CONNECTION_GROUP= LIBNAME option on page 188</a> , <a href="#">DBCONINIT= LIBNAME option on page 200</a> , <a href="#">DBCONTERM= LIBNAME option on page 202</a> , <a href="#">DBLIBTERM= LIBNAME option</a> , <a href="#">DEFER= LIBNAME option</a>                                                                                |

---

## Syntax

**DBLIBINIT=<'>*DBMS-user-command*<'>**

### Syntax Description

#### ***DBMS-user-command***

any DBMS command that the SAS/ACCESS engine can execute and that does not return a result set or output parameters.

## Details

The initialization command that you select can be a script, stored procedure, or any DBMS SQL statement that might provide additional control over the interaction between your SAS/ACCESS interface and the DBMS.

The command executes immediately after the first DBMS connection is successfully established. If the command fails, a disconnection occurs and the libref is not assigned. You must specify the command as a single quoted string unless it is an environment variable.

DBLIBINIT= fails if either CONNECTION=UNIQUE or DEFER=YES, or if both of these LIBNAME options are specified.

When multiple LIBNAME statements share a connection, the initialization command executes only for the first LIBNAME statement, immediately after the DBMS connection is established. (Multiple LIBNAME statements that use CONNECTION=GLOBALREAD and identical values for CONNECTION\_GROUP=, DBCONINIT=, DBCONTERM=, DBLIBINIT=, and DBLIBTERM= options and any DBMS connection options can share the same connection to the DBMS.)

## Example: Allow Only One LIBNAME Statement to Connect

In this example, **CONNECTION=GLOBALREAD** is specified in both LIBNAME statements, but the DBLIBINIT commands are different. Therefore, the second LIBNAME statement fails to share the same physical connection.

```
libname mydblib oracle user=myusr1 pass=mypwd1
  connection=globalread dblibinit='Test';
libname mydblib2 oracle user=myusr1 pass=mypwd1
  connection=globalread dblibinit='NoTest';
```

## DBLIBTERM= LIBNAME Statement Option

Specifies a user-defined termination command to execute once, before the DBMS that is associated with the first connection made by the LIBNAME statement or libref disconnects.

|              |                                                                                                                                                                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                              |
| Category:    | Data Set Control                                                                                                                                                                                                                                                          |
| Default:     | none                                                                                                                                                                                                                                                                      |
| Interaction: | DBLIBTERM= fails if either CONNECTION=UNIQUE or DEFER=YES, or if both of these LIBNAME options are specified.                                                                                                                                                             |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.                                                                                                                                                                                                                                  |

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.  
 Support for Hadoop and JDBC was added in SAS Viya 3.4.  
 Support for Snowflake was added in the August 2019 release of SAS/ACCESS.  
 Support for Spark was added in SAS 9.4M7.  
 Support for Yellowbrick was added in SAS 9.4M7.

See:

[CONNECTION= LIBNAME option](#), [CONNECTION\\_GROUP= LIBNAME option](#),  
[DBCONINIT= LIBNAME option](#), [DBCONTERM= LIBNAME option](#), [DBLIBINIT= LIBNAME option](#), [DEFER= LIBNAME option](#)

## Syntax

**DBLIBTERM=<'>*DBMS-user-command*<'>**

### Syntax Description

***DBMS-user-command***

any DBMS command that can be executed by the SAS/ACCESS engine and that does not return a result set or output parameters.

## Details

The termination command that you select can be a script, stored procedure, or any DBMS SQL statement that might provide additional control over the interaction between the SAS/ACCESS engine and the DBMS. The command executes immediately before SAS terminates the last connection to the DBMS. If the command fails, SAS provides a warning message, but unassigning the library and disconnecting from the DBMS still occur. You must specify the command as a single quoted string.

When two LIBNAME statements share the same physical connection, the termination command is executed only once. CONNECTION=GLOBALREAD and identical values for CONNECTION\_GROUP=, DBCONINIT=, DBCONTERM=, DBLIBINIT=, and DBLIBTERM= options and any DBMS connection options can share the same connection to the DBMS.

## Example: Allow Only One LIBNAME Statement to Connect

In this example, CONNECTION=GLOBALREAD is specified in both LIBNAME statements, but the DBLIBTERM commands are different. Therefore, the second LIBNAME statement fails to share the same physical connection.

```
libname mydblib oracle user=myusr1 pass=mypwd1
  connection=globalread dblibterm='Test';
libname mydblib2 oracle user=myusr1 pass=mypwd1
  connection=globalread dblibterm='NoTest';
```

---

## DBLINK= LIBNAME Statement Option

For Oracle, this option specifies a link from your local database to database objects on another server. For SAP ASE, this option specifies a link from the default database to another database on the server to which you are connected.

|              |                                               |
|--------------|-----------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                  |
| Category:    | Data Set Control                              |
| Default:     | none                                          |
| Data source: | Oracle, SAP ASE                               |
| See:         | <a href="#">DBLARGETABLE= data set option</a> |

---

## Syntax

**DBLINK=***database-link*

---

## Details

**Oracle:** A link is a database object that you use to identify an object stored in a remote database. It contains stored path information. It might also contain user name and password information for connecting to the remote database. If you specify a link, SAS uses it to access remote objects. If you omit this option, SAS accesses objects in only the local database.

**SAP ASE:** This option lets you link to another database within the same server to which you are connected. If you omit this option, SAS can access objects in only your default database.

---

## DBMAX\_TEXT= LIBNAME Statement Option

Determines the length of any very long DBMS character data type, such as BLOB or CLOB, that is read into SAS or written from SAS when using a SAS/ACCESS engine.

|               |                                                                                                                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                |
| Category:     | Data Set Control                                                                                                                                                                                               |
| Alias:        | TEXTSIZE=                                                                                                                                                                                                      |
| Default:      | 1024                                                                                                                                                                                                           |
| Restrictions: | This option applies when you retrieve, append, and update rows in an existing table. It does not apply when you create a table.<br>This option is ignored for CHAR, VARCHAR, and VARCHAR2 (Oracle) data types. |

|               |                                                                                                                                                                                                                                                                                                                                                        |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Requirements: | Set the value to 4000 when you are using procedures that work with SAS High-Performance Analytics Server. [Oracle]<br>Set the value to 4000 or lower when you are using procedures that work with the SAS High-Performance Analytics Server. [Amazon Redshift, DB2, Hadoop, Impala, JDBC, Microsoft SQL Server, ODBC, PostgreSQL, SAP HANA, Spark]     |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, Hadoop, HAWQ, Impala, JDBC, MySQL, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Vertica, Yellowbrick                                                                                                                  |
| Notes:        | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for JDBC was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Spark was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:          | <a href="#">DBMAX_TEXT= data set option</a>                                                                                                                                                                                                                                                                                                            |

## Syntax

**DBMAX\_TEXT=***integer*

## Syntax Description

***integer***

an integer between 1 and 32,767.

## Details

In general, this option applies only to BLOB, CLOB, and other large object or very long DBMS character data types, such as the SAP ASE TEXT data type. Also, remember that the number of bytes that are used to store characters might vary and is based on your session encoding.

If the way that you specify the value of DBMAX\_TEXT= truncates data, the data load fails for that table.

*Hadoop*: This option applies for the STRING data type.

*Oracle*: For SAS 9 or higher, this option applies for CLOB, BLOB, LONG, LONG RAW, and LOB data types. The behavior of the ACCESS and DBLOAD procedures has not changed since SAS 8. So only LONG and LOB data types are valid if you use this option with those procedures.

## DBMSTEMP= LIBNAME Statement Option

Specifies whether SAS creates temporary or permanent tables.

|              |                                                                                                                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                     |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                 |
| Default:     | NO                                                                                                                                                                                                                                                                                               |
| Requirement: | To specify this option, you must first specify CONNECTION=GLOBAL—except for Microsoft SQL Server, which defaults to UNIQUE.                                                                                                                                                                      |
| Interaction: | To access temporary tables, set DBMSTEMP= to YES and set the CONNECTION=LIBNAME option to GLOBAL.                                                                                                                                                                                                |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                   |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Spark was added in SAS 9.4M7.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">CONNECTION= LIBNAME option</a> , <a href="#">Temporary Table Support for SAS/ACCESS</a>                                                                                                                                                                                              |

## Syntax

**DBMSTEMP=**[YES | NO](#)

## Syntax Description

### YES

specifies that SAS creates one or more temporary tables.

### NO

specifies that SAS creates permanent tables.

## Details

To significantly improve performance, you must also specify [DBCOMMIT=0](#). The value for [SCHEMA=](#) is ignored. You can then access and use the DBMS temporary tables using SAS/ACCESS engine librefs that share the global connection that SAS used to create those tables.

To join a temporary table and a permanent table, you need a libref for each table and these librefs must successfully share a global connection.

**Oracle:** Set [INSERTBUFF=1000](#) or higher to significantly improve performance.

**ODBC:** This engine supports DB2, MS SQL Server, or Oracle if you are connected to them.

## Example: Create and Join a Permanent Table and a Temporary Table

This example shows how to use this option to create a permanent and temporary table and then join them in a query. The temporary table might not exist beyond a single PROC step. However, this might not be true for all DBMSs.

```

options sastrace=(,,d,d) nostsuffix sastraceloc=saslog;
LIBNAME permdata DB2 DB=MA40 SCHEMA=SASTDATA connection=global
    dbcommit=0 USER=sasuser PASSWORD=xxx;
LIBNAME tempdata DB2 DB=MA40 SCHEMA=SASTDATA connection=global
    dbcommit=0 dbmstemp=yes USER=sasuser PASSWORD=xxx;
proc sql;
create table tempdata.ptyacc as
(
    select pty.pty_id
    from permdata.pty_rb pty,
        permdata.PTY_ARNG_PROD_RB acc
    where acc.ACC_PD_CTGY_CD = 'LOC'
        and acc.pty_id = pty.pty_id
    group by pty.pty_id having count(*) > 5
);
create table tempdata.ptyacloc as
(
    select ptyacc.pty_id,
        acc.ACC_APPSYS_ID,
        acc.ACC_CO_NO,
        acc.ACCNO,
        acc.ACC_SUB_NO,
        acc.ACC_PD_CTGY_CD
    from tempdata.ptyacc ptyacc,
        perm data.PTY_ARNG_PROD_RB acc
    where ptyacc.pty_id = acc.pty_id
        and acc.ACC_PD_CTGY_CD = 'LOC'
);
create table tempdata.righttab as
(
    select ptyacloc.pty_id
    from permdata.loc_acc loc,
        tempdata.ptyacloc ptyacloc
    where
        ptyacloc.ACC_APPSYS_ID = loc.ACC_APPSYS_ID
        and ptyacloc.ACC_CO_NO = loc.ACC_CO_NO
        and ptyacloc.ACCNO = loc.ACCNO
        and ptyacloc.ACC_SUB_NO = loc.ACC_SUB_NO
        and ptyacloc.ACC_PD_CTGY_CD = loc.ACC_PD_CTGY_CD
        and loc.ACC_CURR_LINE_AM - loc.ACC_LDGR_BL > 20000
);
select * from tempdata.ptyacc
except
select * from tempdata.righttab;
drop table tempdata.ptyacc;
drop table tempdata.ptyacloc;
drop table tempdata.righttab;
quit;

```

## DBNULLKEYS= LIBNAME Statement Option

Controls the format of the WHERE clause when you use the DBKEY= data set option.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Defaults: YES [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP IQ, Snowflake, Vertica, Yellowbrick]  
NO [Informix]

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP IQ, Snowflake, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in SAS Viya 3.4.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

See: [DBKEY= data set option](#), [DBNULLKEYS= data set option](#)

---

## Syntax

**DBNULLKEYS=YES | NO**

### Required Arguments

#### YES

specifies that there might be NULL values in the key columns in a transaction table or a master table.

#### NO

specifies that there are no NULL values in the key columns for a transaction table or a master table.

---

## Details

If there might be NULL values in the transaction table or the master table for the columns that you specify in the [DBKEY=](#) data set option, use **DBNULLKEYS=YES**. This is the default for most interfaces. When you specify **DBNULLKEYS=YES** and also a column that is not specified as NOT NULL in **DBKEY=**, SAS generates a WHERE clause that can find NULL values. For example, if you specify **DBKEY=COLUMN** and **COLUMN** is not specified as NOT NULL, SAS generates a WHERE clause with this syntax:

```
WHERE ((COLUMN = ?) OR ((COLUMN IS NULL) AND (? IS NULL)))
```

With this syntax SAS can prepare the statement once and use it for any (NULL or NOT NULL) value in the column. This syntax can potentially be much less efficient than the shorter form of the WHERE clause below. When you specify DBNULLKEYS=NO or a column that DBKEY= specifies as NOT NULL, SAS generates a simple WHERE clause.

If you know that there are no NULL values in transaction or master tables for the columns that you specify in the DBKEY= option, you can use DBNULLKEYS=NO. This is the default for the Informix interface. If you specify DBNULLKEYS=NO and DBKEY=COLUMN, SAS generates a shorter form of the WHERE clause, regardless of whether the column that is specified in DBKEY= is specified as NOT NULL.

```
WHERE (COLUMN = ?)
```

---

## DBNULLWHERE= LIBNAME Statement Option

Specifies whether character columns in a WHERE clause can contain NULL values.

|              |                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                             |
| Category:    | Data Set Control                                                                                         |
| Default:     | YES                                                                                                      |
| Interaction: | If the DBNULL= data set option is specified, then the value of DBNULLWHERE= is automatically set to YES. |
| Data source: | DB2 under UNIX and PC Hosts, Microsoft SQL Server, ODBC, Oracle, SAP HANA                                |
| Notes:       | Support for this option was added in SAS 9.4M5.<br>Support for this option was added in SAS Viya 3.3.    |
| See:         | <a href="#">DBNULL= data set option</a> , <a href="#">DBNULLWHERE= data set option</a>                   |

---

## Syntax

**DBNULLWHERE=YES | NO**

### Syntax Description

#### **YES**

specifies that there might be a NULL value for a column that is listed in a WHERE clause.

#### **NO**

specifies that none of the columns in a WHERE clause contain NULL values.

---

## Details

This option applies to character columns only.

When DBNULLWHERE=YES, SAS/ACCESS verifies whether blank or NULL values are possible for each character column that you include in a WHERE clause.

When DBNULLWHERE=NO, SAS/ACCESS does not check to see whether NULL values are possible for the specified columns in a WHERE clause. When you know that none of your specified columns contain NULL values, specifying DBNULLWHERE=NO can result in a faster query because fewer conditions are being checked.

## DBPROMPT= LIBNAME Statement Option

Specifies whether SAS displays a window in the SAS windowing environment that prompts the user to enter DBMS connection information before connecting to the DBMS in interactive mode.

|               |                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                                                                                                                                                                                               |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                              |
| Default:      | NO                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Restrictions: | This option is not applicable to SAS Viya.<br>The maximum password length for most of the SAS/ACCESS LIBNAME interfaces is 32 characters.                                                                                                                                                                                                                                                                                     |
| Interaction:  | The DBPROMPT= option interacts with the DEFER=LIBNAME option to determine when the prompt window appears. If DEFER=NO, the DBPROMPT window appears when the LIBNAME statement is executed. If DEFER=YES, the DBPROMPT window appears when you first open a table or view. The DEFER= option normally defaults to NO, but it defaults to YES if DBPROMPT=YES. You can override this default by explicitly specifying DEFER=NO. |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                  |
| Notes:        | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                    |
| See:          | <a href="#">DBPROMPT= data set option</a> , <a href="#">DEFER= LIBNAME option</a>                                                                                                                                                                                                                                                                                                                                             |

## Syntax

**DBPROMPT=YES | NO**

### Syntax Description

#### YES

specifies that SAS displays a window that interactively prompts you for the DBMS connection options the first time the libref is used.

#### NO

specifies that SAS does not display the prompting window.

## Details

If you specify DBPROMPT=YES, it is not necessary to provide connection options with the LIBNAME statement. If you use the LIBNAME statement to specify connection options and DBPROMPT=YES, connection option values are displayed in the window. The value of the password appears as a series of asterisks. You can override all of these values interactively.

The DBPROMPT window usually opens only once for each time that the LIBNAME statement is specified. It might open multiple times if DEFER=YES and the connection fails when SAS tries to open a table. In such cases, the DBPROMPT window appears until a successful connection occurs or you click **Cancel**.

*Oracle:* You can enter 30 characters for the user name and password and up to 70 characters for the path, depending on your platform.

*Teradata:* You can enter up to 30 characters for the user name and password.

---

## Examples

### Example 1: Preventing a Prompt Window from Opening

In this example, the DBPROMPT window does not open when the LIBNAME statement is submitted because DEFER=YES. The DBPROMPT window appears when the PRINT procedure is processed, a connection is made, and the table is opened.

```
libname mydblib oracle dbprompt=yes
      defer=yes;
proc print data=mydblib.staff;
run;
```

### Example 2: Allow a Prompt Window to Open Only Once

In this example, the DBPROMPT window appears while the LIBNAME statement is processing. The DBPROMPT window does not appear in subsequent statements because the DBPROMPT window appears only once per LIBNAME statement.

```
libname mydblib oracle dbprompt=yes
      defer=no;
```

### Example 3: Allow Values to Appear in a Prompt Window

In this example, values provided in the LIBNAME statement are pulled into the DBPROMPT window. The values `myusr1` and `mysrv1` appear in the DBPROMPT window, and the user can edit and confirm them. The password value appears in the DBPROMPT window as a series of asterisks, so the user can also edit it.

```
libname mydblib oracle
  user=myusr1 pw=mypwd1
  path='mysrv1' dbprompt=yes defer=no;
```

## DBSASLABEL= LIBNAME Statement Option

Specifies the column labels an engine uses.

|              |                                                                                                                                                                                                                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                 |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                             |
| Default:     | COMPAT                                                                                                                                                                                                                                                                                                                       |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica                                                            |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Spark was added in SAS 9.4M7.</p> <p>Support for SAP HANA was added in November 2020 (SAS 9.4M7).</p> |
| Tip:         | You can use this option to override the default behavior. It is useful for when PROC SQL uses column labels as headers instead of column aliases.                                                                                                                                                                            |
| See:         | <a href="#">DBSASLABEL= data set option</a>                                                                                                                                                                                                                                                                                  |

## Syntax

**DBSASLABEL=COMPAT | DBMS | NONE**

### Syntax Description

#### **COMPAT**

specifies that the labels returned should be compatible with what the application normally receives—meaning that engines exhibit their normal behavior.

#### **DBMS**

specifies that the engine returns a label exactly as it is stored in the database.

Supports SAP HANA

#### **NONE**

specifies that the engine does not return a column label. The engine returns blanks for the column labels.

## Details

By default, the SAS/ACCESS interface for your DBMS generates column labels from the column names instead of from the real column labels.

## Example: Return Blank Labels for Aliases in Headings

This example shows how to use DBSASLABEL= as a LIBNAME option to return blank column labels so that PROC SQL can use the column aliases as the column headings.

```
libname mylib oracle user=myusr1 pw=mypwd1 dbsaslabel=none;
proc sql;
    select deptno as 'Department ID', loc as Location
        from mylib.dept;
```

Without DBSASLABEL=NONE, aliases are ignored, and DEPTNO and LOC are used as column headings in the result set.

---

## DBSERVER\_ENCODING\_FIXED= LIBNAME Statement Option

Specifies whether Oracle database encoding is a fixed width.

|              |                                                                              |
|--------------|------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                 |
| Category:    | Data Set Control                                                             |
| Default:     | YES or NO, based on Oracle server encoding                                   |
| Restriction: | Applies only when you read Oracle tables in SAS                              |
| Interaction: | To avoid truncation issues, use care when setting this option to YES.        |
| Supports:    | NLS                                                                          |
| Data source: | Oracle                                                                       |
| Note:        | Support for this option was added in SAS 9.4M1.                              |
| See:         | <a href="#">SAS/ACCESS LIBNAME options for NLS</a>                           |
| CAUTION:     | <b>To avoid truncation issues, use care when setting this option to YES.</b> |

---

## Syntax

**DBSERVER\_ENCODING\_FIXED=YES | NO**

## Syntax Description

### YES

indicates that database table column lengths in characters are a fixed width. Use this value to adjust byte lengths within SAS for any database column lengths that are specified in bytes. The number of characters is calculated by dividing the byte length by the value in `DBSERVER_MAX_BYTES=`.

### NO

indicates that database table column lengths are not a fixed width.

## Example: Specify Oracle Database Encoding

For this example, the SAS session encoding is euc-cn, the locale is Chinese, and the Oracle server encoding is UTF8.

```
/* Read */
/* Prepare in Oracle a table with char(6), named test. */
/* See from SQLPLUS: */
SQL> desc test;
```

| Name | Null? | Type    |
|------|-------|---------|
| ID   |       | CHAR(6) |

```
libname lib1 oracle path=nlsbip08 user=myusr1 pw=mypwd1
      dbserver_max_bytes=3 dbclient_max_bytes=2
      dbserver_encoding_fixed = yes;

proc contents data=lib1.test;run;

libname lib2 oracle path=nlsbip08 user=myusr1 pw=mypwd1
      dbserver_max_bytes=3 dbclient_max_bytes=2
      dbserver_encoding_fixed=no;

proc contents data=lib2.test;run;
```

| Result                                      |          |      |     |        |          |       |
|---------------------------------------------|----------|------|-----|--------|----------|-------|
| First Proc contents                         |          |      |     |        |          |       |
| Alphabetic List of Variables and Attributes |          |      |     |        |          |       |
| #                                           | Variable | Type | Len | Format | Informat | Label |
| 1                                           | ID       | Char | 4   | \$4.   | \$4.     | ID    |
| Second Proc contents                        |          |      |     |        |          |       |
| Alphabetic List of Variables and Attributes |          |      |     |        |          |       |
| #                                           | Variable | Type | Len | Format | Informat | Label |
| 1                                           | ID       | Char | 12  | \$12.  | \$12.    | ID    |

---

## DBSERVER\_MAX\_BYTES= LIBNAME Statement Option

Specifies the maximum number of bytes per single character in the database server encoding.

|              |                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                   |
| Category:    | Data Set Control                                                                                                                                                                                               |
| Alias:       | DB_MAX_BYT[ES]= [Impala]                                                                                                                                                                                       |
| Defaults:    | 0 [Impala, Snowflake]<br>usually 1 [Oracle, SAP ASE, Vertica]<br>none [Amazon Redshift, DB2 under UNIX and PC Hosts, Yellowbrick]                                                                              |
| Supports:    | NLS                                                                                                                                                                                                            |
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, Impala, Oracle, SAP ASE, Snowflake, Vertica, Yellowbrick                                                                                                         |
| Notes:       | Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">DBCLIENT_MAX_BYTES= LIBNAME option</a> , <a href="#">SAS/ACCESS LIBNAME options for NLS</a>                                                                                                        |

---

## Syntax

**DBSERVER\_MAX\_BYTES=*max-server-bytes***

---

## Details

Use this option to derive (adjust the value of) the number of characters from the client column lengths that byte semantics initially creates. Although the default is usually 1, you can use this option to set it to another value if this information is available from the external data source server.

**SAP ASE:** You can use this option to specify different byte encoding between the SAS client and the SAP ASE server. For example, if the client uses double-byte encoding and the server uses multibyte encoding, specify DBSERVER\_MAX\_BYT[ES]=3. In this case, the SAS/ACCESS engine evaluates this option only if you specify a value that is greater than 2. Otherwise, it indicates that both client and server use the same encoding scheme.

## Examples

### Example 1: Adjust Specific Column Lengths

Only the lengths that you specify with DBSERVER\_MAX\_BYTES= affect column lengths that byte semantics created initially.

```
libname x4 &engine &connopt DBSERVER_MAX_BYTES=4
DBCLIENT_MAX_BYTES=1 ADJUST_NCHAR_COLUMN_LENGTHS=no;
proc contents data=x4.char_sem; run;
proc contents data=x4.nchar_sem; run;
proc contents data=x4.byte_sem; run;
proc contents data=x4.mixed_sem; run;
```

### Example 2: Specify Different Settings for Various Options(#1)

```
libname x5 &engine &connopt ADJUST_NCHAR_COLUMN_LENGTHS=NO
ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=NO DBCLIENT_MAX_BYTES=3;
proc contents data=x5.char_sem; run;
proc contents data=x5.nchar_sem; run;
proc contents data=x5.byte_sem; run;
proc contents data=x5.mixed_sem; run;
```

### Example 3: Specify Different Settings for Various Options (#2)

```
libname x6 &engine &connopt ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS=YES
ADJUST_NCHAR_COLUMN_LENGTHS=YES DBCLIENT_MAX_BYTES=3;
proc contents data=x6.char_sem; run;
proc contents data=x6.nchar_sem; run;
proc contents data=x6.byte_sem; run;
proc contents data=x6.mixed_sem; run;
```

---

## DBSLICEPARM= LIBNAME Statement Option

Controls the scope of DBMS threaded Reads and the number of threads.

|           |                                                                                                                                                                                                                                                    |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement (also available as a SAS configuration option, SAS invocation option, global SAS option, or data set option)                                                                                                          |
| Category: | Data Set Control                                                                                                                                                                                                                                   |
| Defaults: | NONE [DB2 under UNIX and PC Hosts, Greenplum, Microsoft SQL Server, Vertica]<br>THREADED_APPS, none [HAWQ]<br>THREADED_APPS,2 [DB2 under z/OS, Oracle, Teradata]<br>THREADED_APPS,2 or THREADED_APPS,3 [Informix, ODBC, SAP ASE, SAP HANA, SAP IQ] |

|              |                                                                                                                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Informix, Microsoft SQL Server, ODBC, Oracle, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica                                                                                                                                                     |
| Notes:       | Support for Vertica was added for SAS 9.4.<br>Support for SAP HANA was added in SAS 9.4M1.<br>Support for Greenplum was added in SAS 9.4M2.<br>Support for HAWQ was added in SAS 9.4M3.                                                                                                                      |
| See:         | <a href="#">DBSLICE= data set option</a> , <a href="#">DBSLICEPARM= data set option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">SLEEP= LIBNAME option</a> , <a href="#">SLEEP= data set option</a> , <a href="#">TENACITY= LIBNAME option</a> , <a href="#">TENACITY= data set option</a> |

## Syntax

**DBSLICEPARM=NONE | THREADED\_APPS | ALL**  
**DBSLICEPARM=( NONE | THREADED\_APPS | ALL < *max-threads* > )**  
**DBSLICEPARM=( NONE | THREADED\_APPS | ALL < , *max-threads* > )**

## Syntax Description

### **NONE**

disables DBMS threaded Read. SAS reads tables on a single DBMS connection, as it did with SAS 8 and earlier.

### **THREADED\_APPS**

makes fully threaded SAS procedures (threaded applications) eligible for threaded Reads.

### **ALL**

makes all read-only librefs eligible for threaded Reads. This includes SAS threaded applications, as well as the SAS DATA step and numerous SAS procedures.

### ***max-threads***

a positive integer value that specifies the maximum number of connections per table read. The second parameter of the option determines the number of threads to read the table in parallel. The number of partitions on the table determine the number of connections made to the Oracle server for retrieving rows from the table. A partition or portion of the data is read on each connection. The combined rows across all partitions are the same regardless of the number of connections. That is, changes to the number of connections do not change the result set. Increasing the number of connections instead redistributes the same result set across more connections.

If the database table is not partitioned, SAS creates *max-threads* number of connections with *WHERE MOD()...* predicates and the same number of threads.

There are diminishing returns when increasing the number of connections. With each additional connection, more burden is placed on the DBMS, and a smaller percentage of time saved on the SAS step. See the DBMS-specific reference section for details about partitioned reads before using this parameter.

## Details

You can use DBSLICEPARM= in numerous locations. The usual rules of option precedence apply: A table option has the highest precedence, then a LIBNAME option, and so on. SAS configuration file option has the lowest precedence because DBSLICEPARM= in any of the other locations overrides that configuration value.

DBSLICEPARM=ALL and DBSLICEPARM=THREADED\_APPS make SAS programs eligible for threaded Reads. To see whether threaded Reads are actually generated, turn on SAS tracing and run a program, as shown in this example.

```
options sastrace=",t" sastraceloc=saslog nostsuffix;
proc print data=lib.dbtable(dbsliceparm=(ALL));
  where dbcol>1000;
run;
```

If you want to directly control the threading behavior, use the DBSLICE= data set option.

*Greenplum, HAWQ:* There is no default value for the maximum number of connections per table read. This value depends on the number of partitions in a table and on arguments that are used with the MOD function in a WHERE clause. For more information, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page [76](#).

DB2 under UNIX and PC Hosts, Informix, Microsoft SQL Server, ODBC, SAP ASE, SAP IQ: The default thread number depends on whether an application passes in the number of threads (CPUCOUNT=) and whether the data type of the column that was selected for data partitioning is binary.

For more information about autopartitioning with DB2 under z/OS, see “[READBUFF= Restriction](#)” on page [798](#).

---

## Examples

### Example 1: Disable Threaded Read for All SAS Users

Here is how to use DBSLICEPARM= in a SAS configuration file entry in Windows to turn off threaded Reads for all SAS users.

```
-dbsliceparm NONE
```

### Example 2: Enable Threaded Reads for Read-Only References

Here is how you can use DBSLICEPARM= as a z/OS invocation option to turn on threaded Reads for read-only references to DBMS tables throughout a SAS job.

```
sas o(dbsliceparm=ALL)
```

## Example 3: Increase Maximum Threads (as a SAS Global Option)

In this example, you can use DBSLICEPARM= as a SAS global option to increase maximum threads to three for SAS threaded applications. This OPTIONS statement is typically one of the first statements in your SAS code.

```
options dbsliceparm=(threaded_apps,3);
```

## Example 4: Enable Threaded Reads for References Using a Particular Libref

You can use DBSLICEPARM= as a LIBNAME option to turn on threaded Reads for read-only table references that use this particular libref, as shown in this example.

```
libname dblib oracle user=myusr1 password=mypwd1 dbsliceparm=ALL;
```

## Example 5: Enable Threaded Reads as a Table-Level Option

Here is how to use DBSLICEPARM= as a table-level option to turn on threaded Reads for this particular table, requesting up to four connections.

```
proc reg SIMPLE;
  data=dblib.customers (dbsliceparm=(all,4));
  var age weight;
  where years_active>1;
run;
```

# DEFAULT\_AUTH\_PLUGIN= LIBNAME Statement Option

Specifies the authentication plugin to use for a connection to a MySQL compatible data source.

|              |                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                |
| Category:    | Data Access                                                                                                 |
| Default:     | caching_sha2_password (for MySQL 8 and higher client)                                                       |
| Requirement: | When you use a MySQL 8 client to connect to SingleStore, specify DEFAULT_AUTH_PLUGIN=mysql_native_password. |
| Data source: | MySQL                                                                                                       |

## Syntax

**DEFAULT\_AUTH\_PLUGIN=caching\_sha2\_password | mysql\_native\_password**

## Required Argument

### caching\_sha2\_password | mysql\_native\_password

specifies the authentication plugin to use for a connection to a MySQL compatible data source.

## DEFER= LIBNAME Statement Option

Specifies when the connection to the DBMS occurs.

Valid in: SAS/ACCESS LIBNAME statement, CONNECT statement

Category: Data Access

Default: NO

Restriction: Do not set DEFER= to YES when you are using PROC DS2 to interact with your data. PROC DS2 assumes that a connection to your database is in place when the procedure is invoked.

Interactions: The default value of NO is overridden if DBPROMPT=YES.

The DEFER= option is ignored when CONNECTION=UNIQUE because a connection is performed every time a table is opened.

*Microsoft SQL Server, Netezza, ODBC:* When you specify DEFER=YES, you must also specify the [PRESERVE\\_TAB\\_NAMES=](#) and [PRESERVE\\_COL\\_NAMES=](#) options to the values that you want. Normally, SAS queries the data source to determine the correct defaults for these options during LIBNAME assignment, but specifying DEFER=YES postpones the connection. Because these values must be specified at the time of LIBNAME assignment, you must assign them explicitly when you specify DEFER=YES.

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for JDBC was added in SAS Viya 3.4.

Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

See: [CONNECTION= LIBNAME option](#), [DBPROMPT= LIBNAME option](#)

## Syntax

**DEFER=YES | NO**

## Syntax Description

### **YES**

specifies that the connection to the DBMS occurs when a table in the DBMS is opened.

### **NO**

specifies that the connection to the DBMS occurs when the libref is assigned by a LIBNAME statement.

## DEGREE= LIBNAME Statement Option

Determines whether DB2 uses parallelism.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: ANY

Data source: DB2 under z/OS

See: [DEGREE= data set option](#)

## Syntax

**DEGREE=ANY | 1**

## Syntax Description

### **ANY**

enables DB2 to use parallelism, and issues the SET CURRENT DEGREE ='xxx' for all DB2 threads that use that libref.

### **1**

explicitly disables the use of parallelism.

## Details

When DEGREE=ANY, DB2 has the option of using parallelism, when it is appropriate.

Specifying DEGREE=1 prevents DB2 from performing parallel operations. Instead, DB2 is restricted to performing one task that, although this is perhaps slower, it uses less system resources.

---

## DELETE\_MULT\_ROWS= LIBNAME Statement Option

Indicates whether to let SAS delete multiple rows from a data source, such as a DBMS table.

|              |                                                                                                                                                                                                                                                                                                                                    |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                       |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                   |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, Greenplum, HAWQ, Impala, Microsoft SQL Server, ODBC, OLE DB, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick                                                                                                                                                                                 |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Impala was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">UPDATE_MULT_ROWS= LIBNAME option</a>                                                                                                                                                                                                                                                                                   |

---

## Syntax

**DELETE\_MULT\_ROWS=**[YES](#) | NO

### Syntax Description

#### YES

specifies that SAS/ACCESS processing continues if multiple rows are deleted.  
This might produce unexpected results.

#### NO

specifies that SAS/ACCESS processing does not continue if multiple rows are deleted.

---

## Details

Some providers do not handle this DBMS SQL statement well and therefore delete more than the current row:

DELETE...WHERE CURRENT OF CURSOR

---

## DIMENSION= LIBNAME Statement Option

Specifies whether the database creates dimension tables or fact tables.

|              |                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                |
| Category:    | Data Set Control                                                                                                                            |
| Default:     | NO                                                                                                                                          |
| Data source: | Aster                                                                                                                                       |
| See:         | <a href="#">DIMENSION= data set option</a> , <a href="#">PARTITION_KEY= LIBNAME option</a> , <a href="#">PARTITION_KEY= data set option</a> |

---

## Syntax

**DIMENSION=YES | NO**

### Syntax Description

#### **YES**

specifies that the database creates dimension tables.

#### **NO**

specifies that the database creates fact tables.

---

## DIRECT\_EXE= LIBNAME Statement Option

Lets an SQL DELETE statement be passed directly to a DBMS with pass-through.

|              |                                                                                                                                                                                                                      |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                         |
| Category:    | Data Set Control                                                                                                                                                                                                     |
| Default:     | none                                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.                                                                                           |
| Tip:         | Performance improves significantly by using DIRECT_EXE=. This is because the SQL delete statement is passed directly to the DBMS instead of having SAS read the entire result set and delete one row at a time.      |
| See:         | <a href="#">DBIDIRECTEXEC</a> system option                                                                                                                                                                          |

## Syntax

**DIRECT\_EXE=DELETE**

### Syntax Description

#### **DELETE**

specifies that an SQL DELETE statement is passed directly to the DBMS for processing.

### Details

Use the DBIDIRECTEXEC system option to pass additional statements directly to the database.

### Example: Empty a Table from a Database

```
libname x oracle user=myusr1 password=mypwd1
      path=oraclev8 schema=testschema
      direct_exe=delete; /* Create an Oracle table of 5 rows. */
      data x.dbi_dft;
      do col1=1 to 5;
      output;
      end;
      run;
      options sastrace=",,d" sastraceloc=saslog nostsuffix;
      proc sql;
      delete * from x.dbi_dft;
      quit;
```

By turning trace on, you should see something similar to this:

*Output 12.1 SAS Log Output*

ORACLE\_9: Executed:  
delete from dbi\_dft

## DIRECT\_SQL= LIBNAME Statement Option

Specifies whether generated SQL is passed to the DBMS for processing.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: YES

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                  |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">SQL_FUNCTIONS= LIBNAME option</a>                                                                                                                                                                                                                                                                                                        |

## Syntax

**DIRECT\_SQL=NO | NOFUNCTIONS | NONE | ONCE**  
**DIRECT\_SQL=NOGENSQL | NONE | NOWHERE | NOMULTOUTJOINS**

## Syntax Description

### **NO**

specifies that generated SQL from PROC SQL is not passed to the DBMS for processing. This is the same as specifying the value NOGENSQL.

### **NOFUNCTIONS**

prevents SQL statements from being passed to the DBMS for processing when they contain functions.

### **NOGENSQL**

prevents PROC SQL from generating SQL to be passed to the DBMS for processing.

### **NONE**

specifies that generated SQL is not passed to the DBMS for processing. This includes SQL that is generated from PROC SQL, SAS functions that can be converted into DBMS functions, joins, and WHERE clauses.

### **NOWHERE**

prevents WHERE clauses from being passed to the DBMS for processing. This includes SAS WHERE clauses and PROC SQL generated or PROC SQL specified WHERE clauses.

### **NOMULTOUTJOINS**

specifies that PROC SQL does not attempt to pass any multiple outer joins to the DBMS for processing. Other join statements might be passed down, however, including portions of a multiple outer join.

### **ONCE**

specifies that PROC SQL passes generated SQL to the DBMS for processing. PROC SQL tries only once, however. It does not try again if the first attempt fails.

### **YES**

specifies that generated SQL from PROC SQL is passed directly to the DBMS for processing.

## Details

By default, processing is passed to the DBMS whenever possible. The database is generally able to process the functionality more efficiently than SAS. In some instances, however, you might not want the DBMS to process the SQL. For example, the presence of null values in the DBMS data might cause different results depending on whether the processing takes place in SAS or in the DBMS. If you do not want the DBMS to handle the SQL, use DIRECT\_SQL= to force SAS to handle some or all SQL processing.

If you specify DIRECT\_SQL=NOGENSQL, PROC SQL does not generate DBMS SQL. This means that SAS functions, joins, and DISTINCT processing that occur *within* PROC SQL are not passed to the DBMS for processing. (SAS functions *outside* PROC SQL can still be passed to the DBMS.) However, if PROC SQL contains a WHERE clause, the WHERE clause *is* passed to the DBMS, if possible. Unless you specify DIRECT\_SQL=NOWHERE, SAS attempts to pass all WHERE clauses to the DBMS.

If you specify more than one value for this option, separate the values with spaces and enclose the list of values in parentheses. For example, you could specify DIRECT\_SQL=(NOFUNCTIONS NOWHERE).

DIRECT\_SQL= overrides the [SQL\\_FUNCTIONS= LIBNAME](#) option. If you specify SQL\_FUNCTIONS=ALL and DIRECT\_SQL=NONE, no functions are passed.

---

## Examples

### Example 1: Prevent a DBMS from Processing a Join

This example prevents the DBMS from processing a join between two tables by setting DIRECT\_SQL=NOGENSQL. SAS processes the join instead.

```
proc sql;
create view work.v as
  select tab1.deptno, dname from
    mydblib.table1 tab1,
    mydblib.table2 tab2
  where tab1.deptno=tab2.deptno
  using libname mydblib oracle user=myusr1
    password=mypwd1 path=mysrv1 direct_sql=nogensql;
```

### Example 2: Prevent a DBMS from Processing a SAS Function

```
libname mydblib oracle user=myusr1 password=mypwd1 direct_sql=nofunctions;
proc print data=mydblib.tab1;
  where lastname=soundex ('Paul');
```

---

## DRIVER\_TRACE= LIBNAME Statement Option

Requests tracing information that logs transaction records to an external file.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: none

Data source: Google BigQuery

Note: Support for this option was added in the August 2019 release of SAS/ACCESS.

See: [DRIVER\\_TRACEFILE= LIBNAME option](#), [DRIVER\\_TRACEOPTIONS= LIBNAME option](#)

---

## Syntax

**DRIVER\_TRACE=***tracing-level*

### Syntax Description

#### ***tracing-level***

specifies the level of information that is recorded in the log file.

You can specify multiple values within parentheses, separated by a comma. For example, you can specify SQL and DRIVER values as `DRIVER_TRACE=(SQL, DRIVER)`.

Here are the possible values:

ALL       records the information from all of the possible values of `DRIVER_TRACE=`.

SQL       records the SQL statements that are sent to the DBMS in the trace log.

DRIVER    records driver-specific information in the trace log.

---

## Details

The log file for the `DRIVER_TRACE=` option is specified by the `DRIVER_TRACEFILE= LIBNAME` option.

---

## DRIVER\_TRACEFILE= LIBNAME Statement Option

Specifies the name of the trace log.

|               |                                                                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                |
| Category:     | Data Set Control                                                                                                                                                            |
| Default:      | none                                                                                                                                                                        |
| Interactions: | This option is required when you specify the DRIVER_TRACE= LIBNAME option.<br>(Optional) You can control trace log formatting with the DRIVER_TRACEOPTIONS= LIBNAME option. |
| Data source:  | Google BigQuery                                                                                                                                                             |
| Note:         | Support for this option was added in the August 2019 release of SAS/ACCESS.                                                                                                 |
| See:          | <a href="#">DRIVER_TRACE= LIBNAME option</a> , <a href="#">DRIVER_TRACEOPTIONS= LIBNAME option</a>                                                                          |

---

## Syntax

**DRIVER\_TRACEFILE='path-and-filename'**

### Syntax Description

***path-and-filename***

specifies the path and filename for the trace log. Include the path and filename within single or double quotation marks.

---

## DRIVER\_TRACEOPTIONS= LIBNAME Statement Option

Specifies options for the trace log file.

|              |                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                    |
| Category:    | Data Set Control                                                                                |
| Default:     | The trace log is overwritten and includes no time stamps or thread identification.              |
| Data source: | Google BigQuery                                                                                 |
| Note:        | Support for this option was added in the August 2019 release of SAS/ACCESS.                     |
| See:         | <a href="#">DRIVER_TRACE= LIBNAME option</a> , <a href="#">DRIVER_TRACEFILE= LIBNAME option</a> |

---

## Syntax

**DRIVER\_TRACEOPTIONS**=*value*

### Syntax Description

***value***

specifies options to control formatting and other properties for the trace log.

Here are the possible values:

|             |                                                               |
|-------------|---------------------------------------------------------------|
| APPEND      | adds tracing information to the end of an existing trace log. |
| TIMESTAMP   | prepends each line of the trace log with a time stamp.        |
| THREADSTAMP | prepends each line of the trace log with a thread identifier. |

---

## DRIVER\_VENDOR= LIBNAME Statement Option

Specifies the name of the ODBC driver vendor.

|              |                                                                                                                                                                             |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                             |
| Category:    | Data Set Control                                                                                                                                                            |
| Default:     | CLOUDERA                                                                                                                                                                    |
| Interaction: | Use the SAS_IMPALA_DRIVER_VENDOR environment variable for the entire SAS session. When both are specified, the LIBNAME option has precedence over the environment variable. |
| Data source: | Impala                                                                                                                                                                      |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M3.                                                                                                                     |
| Examples:    | Specify the LIBNAME option:                                                                                                                                                 |

```
libname imp impala user=myuser1 password=mypwd1 server=myimpalaserver
      schema=myschema driver_vendor=cloudera;
```

Specify the environment variable on PC hosts:

```
SAS_IMPALA_DRIVER_VENDOR CLOUDERA
```

Export the environment variable on UNIX hosts:

```
export SAS_IMPALA_DRIVER_VENDOR=CLOUDERA
```

Specify the environment variable at SAS invocation for PC and UNIX:

```
sas -set SAS_IMPALA_DRIVER_VENDOR CLOUDERA
```

---

## Syntax

**DRIVER\_VENDOR**= <CLOUDERA> | <DATADIRECT> | <MAPR> | <PROGRESS>

## Optional Arguments

### **CLOUDERA**

specifies Cloudera as the ODBC driver vendor.

### **DATADIRECT**

specifies DataDirect as the ODBC driver vendor.

### **MAPR**

specifies MapR as the ODBC driver vendor.

### **PROGRESS**

specifies Progress as the ODBC driver vendor.

---

## ENABLE\_BULK= LIBNAME Statement Option

Lets the connection process bulk copy when loading data into an SAP ASE table.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: YES

Data source: SAP ASE

Note: In SAS 7 and previous releases, this option was called BULKCOPY=. In SAS 8 and later, an error is returned if you specify BULKCOPY=.

See: [BULK\\_BUFFER= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**ENABLE\_BULK=NO | YES**

### Syntax Description

#### **NO**

disables bulk copy ability for the libref.

#### **YES**

lets the connection perform bulk copy of SAS data into SAP ASE.

---

## Details

Bulk copy groups rows so that they are inserted as a unit into the SAP ASE table. Using bulk copy can improve performance.

If you use both the, **ENABLE\_BULK= LIBNAME** option and the [BULKLOAD= data set option](#), values for both options must be the same or an error is returned. However, because **ENABLE\_BULK=YES** is the default value, you do not need to specify **ENABLE\_BULK=** to use the [BULKLOAD= data set option](#).

---

## ERRLIMIT= LIBNAME Statement Option

Specifies the number of errors that are allowed while using the TPT or Fastload utility before SAS stops loading data to Teradata.

Valid in: DATA and PROC steps (wherever TPT or Fastload is used)

Category: Data Set Control

Default: none

Data source: Teradata

See: [ERRLIMIT= data set option](#), [DBCOMMIT= LIBNAME option](#), [DBCOMMIT= data set option](#), [ML\\_CHECKPOINT= data set option](#)

---

## Syntax

**ERRLIMIT=***integer*

### Syntax Description

***integer***

specifies a positive integer that represents the number of errors after which SAS stops loading data.

---

## Details

SAS stops loading data when it reaches the specified number of errors and Fastload pauses. When Fastload pauses, you cannot use the table that is being loaded. Restart capability for Fastload is not yet supported, so you must manually delete the error tables before SAS can reload the table.

This option applies to TPT Load, Update, and Stream operations. For TPT, this option sets the value for TD\_ERROR\_LIMIT.

---

## Example

In this example, SAS stops processing and pauses Fastload when it encounters the 10th error.

```
libname mydblib teradata user=terouser pw=XXXXXX ERRLIMIT=10;
data mydblib.trfload(bulkload=yes dbtype=(i='int check (i > 11)' ));
do
  i=1 to 50000;output;
end;
run;
```

---

## ESCAPE\_BACKSLASH= LIBNAME Statement Option

Specifies whether backslashes in literals are preserved during data copy from a SAS data set to a table.

|              |                                                   |
|--------------|---------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                      |
| Category:    | Data Set Control                                  |
| Default:     | NO                                                |
| Data source: | MySQL                                             |
| See:         | <a href="#">ESCAPE_BACKSLASH= data set option</a> |

---

## Syntax

**ESCAPE\_BACKSLASH=YES | NO**

### Syntax Description

#### YES

specifies that an additional backslash is inserted in every literal value that already contains a backslash.

#### NO

specifies that backslashes that exist in literal values are not preserved. An error results.

---

## Details

MySQL uses the backslash as an escape character. When data that is copied from a SAS data set to a MySQL table contains backslashes in literal values, the MySQL interface can preserve these if ESCAPE\_BACKSLASH=YES.

---

## FASTEXPORT= LIBNAME Statement Option

Specifies whether the SAS/ACCESS engine uses the TPT API to read data.

|              |                              |
|--------------|------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement |
| Category:    | Bulk Loading                 |
| Default:     | NO                           |
| Data source: | Teradata                     |

See:

BULKLOAD= LIBNAME option, BULKLOAD= data set option, DBSLICEPARM= LIBNAME option, DBSLICEPARM= data set option, DBSLICEPARM= system option, FASTLOAD=LIBNAME option, LOGDB= LIBNAME option, Maximizing Teradata Load Performance, MULTILOAD= data set option, QUERY\_BAND= LIBNAME option, QUERY\_BAND= data set option, SLEEP= LIBNAME option, SLEEP= data set option, TENACITY= LIBNAME option, TENACITY= data set option, TPT\_BLOCK\_SIZE= data set option, TPT\_DATA\_ENCRYPTION= data set option, TPT\_LOG\_TABLE= data set option, TPT\_MAX\_SESSIONS= LIBNAME option, TPT\_MAX\_SESSIONS= data set option, TPT\_MIN\_SESSIONS= data set option TPT\_TRACE\_LEVEL= data set option, TPT\_TRACE\_LEVEL\_INF= data set option, TPT\_UNICODE\_PASSTHRU= LIBNAME option

## Syntax

**FASTEXPORT=YES | NO**

### Syntax Description

#### YES

specifies that data is loaded from Teradata into SAS using the TPT Export driver.

#### NO

specifies that the TPT Export driver is not to be used.

## Details

By using the TPT API, you can read data from a Teradata table without working directly with the stand-alone Teradata FastExport utility. When FASTEXPORT=YES, SAS uses the TPT API export driver for bulk reads. If SAS cannot use the TPT API —because of an error or because it is not installed on the system—SAS still tries to read the data. However, no error is produced. To see whether SAS used the TPT API to read data, look for this message in the SAS log:

NOTE: Teradata connection: TPT FastExport has read n row(s).

When you specify a query band on this option, you must specify the **DBSLICEPARM=LIBNAME** option. The query band is passed as a SESSION query band to the FastExport utility.

To see whether threaded Reads are actually generated, turn on SAS tracing by specifying **OPTIONS SASTRACE=" , , , d"**; in your program.

## Example

In this example, the TPT API reads SAS data from a Teradata table. SAS still tries to read data even if it cannot use the TPT API.

```
Libname tera Teradata user=myusr1 pw=mypwd1 FASTEXPORT=YES;
```

```

/* Create data */
Data teratestdata;
Do i=1 to 100;
   Output;
End;
Run;
/* Read using FastExport TPT. This note appears in the SAS log if SAS uses TPT.
NOTE: Teradata connection: TPT FastExport has read

n row(s).*/
Data worktestdata;
Set teratestdata;
Run;

```

---

## FASTLOAD= LIBNAME Statement Option

Specifies whether to use the TPT load driver.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Alias:       | BULKLOAD=                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">LOGDB= LIBNAME option</a> , <a href="#">FASTEXPORT= LIBNAME option</a> , <a href="#">FASTLOAD= data set option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">QUERY_BAND= LIBNAME option</a> , <a href="#">QUERY_BAND= data set option</a> , <a href="#">SLEEP= LIBNAME option</a> , <a href="#">SLEEP= data set option</a> , <a href="#">TENACITY= LIBNAME option</a> , <a href="#">TPT= LIBNAME option</a> , <a href="#">TENACITY= data set option</a> , <a href="#">TPT_DATA_ENCRYPTION= data set option</a> , <a href="#">TPT_DATA_ENCRYPTION= LIBNAME option</a> , <a href="#">TPT_UNICODE_PASSTHRU= LIBNAME option</a> |

---

## Syntax

**FASTLOAD YES | NO**

### Syntax Description

#### YES

specifies that the SAS/ACCESS engine uses the TPT load driver to load the data.

#### NO

specifies that the SAS/ACCESS engine does not use the TPT load driver to load the data.

## Details

### Default Behavior for Data Transfer

By default for Teradata, you transfer data by using the Teradata Parallel Transporter (TPT) API. FastLoad is considered a legacy feature. To enable data transfer using the TPT API, enable the [TPT= LIBNAME option](#) or the [TPT= data set option](#). For more information, see “[Maximizing Teradata Load and Read Performance](#)” on page [1337](#).

### Best Practice for Specifying FASTLOAD=

You can specify FASTLOAD= using a data set option or LIBNAME option. Use care with the FASTLOAD= LIBNAME option (rather than the data set option). Anytime you insert data from SAS into Teradata tables, the FastLoad protocol is used. This uses Teradata utility slots and might also cause other load jobs to fail.

Specifying the FASTLOAD= LIBNAME option on a SAS library hides it from users. In a BI environment, it can be difficult to tell whether the option is specified. If you are setting up SAS libraries that are used only by ETL jobs, then it might be acceptable to use the LIBNAME option.

---

**Note:** A best practice recommendation is to use [FASTLOAD=](#) as a data set option unless you have a compelling reason to use it as a LIBNAME option.

---

```

libname mytera TERADATA server=teraserv user=bob pw=bob1;
/* Create and load a table using a DATA step. SAS numeric is */
/* forced to be an INTEGER. */
data mytera.table0 (FASTLOAD=YES DBTYPE= (x= INTEGER));
do x = 1 to 1000000;
    output;
end; run;

/* Load an existing table using PROC APPEND. The table must meet */
/* certain requirements. */
PROC SQL;
    CONNECT TO TERADATA (USER=bob PW=bob1 SERVER=tera5500);
    EXECUTE (DROP TABLE loadThisTable) by TERADATA;
    EXECUTE (COMMIT) BY TERADATA;
    EXECUTE (CREATE MULTISET TABLE loadThisTable ( a INTEGER , b
CHAR(10)
        PRIMARY INDEX (a)) by TERADATA;
    EXECUTE (COMMIT) BY TERADATA;
QUIT;
DATA work.loadData;
    FORMAT b $10.;
    a = 1;
    output;
    b = 'One';
    output;
    a = 2;
    output;
    b = 'Two';

```

```

        output;
a = 3;
        output;
b = 'Three';
        output;
RUN;

libname mytera teradata server=teraserv user=bob pw=bob1 FASTLOAD=YES;
PROC APPEND BASE=mytera.loadThisTable (BL_LOG=BOB_APPEND_ERR)
    DATA=work.loadData;
RUN;

```

## FETCH\_IDENTITY= LIBNAME Statement Option

Returns the value of the last inserted identity value.

Valid in: SAS/ACCESS LIBNAME statement  
 Category: Data Set Control  
 Default: NO  
 Data source: DB2 under UNIX and PC Hosts  
 See: [FETCH\\_IDENTITY= data set option](#)

## Syntax

**FETCH\_IDENTITY=YES | NO**

### Syntax Description

#### YES

returns the value of the last inserted identity value.

#### NO

disables this option.

## Details

You can use this option instead of issuing a separate SELECT statement after an INSERT statement. If FETCH\_IDENTITY=YES and the INSERT that is executed is a single-row INSERT, the engine calls the DB/2 identity\_val\_local function and places the results into the SYSDB2\_LAST\_IDENTITY macro variable. Because the DB2 engine default is multirow inserts, you must set [INSERTBUFF=1](#) to force a single-row INSERT.

---

## HDFS\_PRINCIPAL= LIBNAME Option LIBNAME Statement Option

Specifies the Kerberos principal for HDFS.

|              |                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                         |
| Category:    | Data Access                                                                                                                          |
| Alias:       | HDFS_KERBEROS_PRINCIPAL=                                                                                                             |
| Default:     | none                                                                                                                                 |
| Restriction: | The HDFS_PRINCIPAL= LIBNAME option applies only when BULKLOAD=YES and only when you configure HDFS to allow Kerberos authentication. |
| Data source: | Impala                                                                                                                               |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M2.                                                                              |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">IMPALA_PRINCIPAL= LIBNAME option</a>                                          |

---

## Syntax

**HDFS\_PRINCIPAL='*principal*'**

### Required Argument

***principal***

specifies the server's Kerberos service principal name (SPN). Surround the principal value with single or double quotation marks. For example, you might specify a principal value that is similar to the following code:

```
hdfs_principal='hdfs/hdfs_host.example.com@TEST.EXAMPLE.COM'
```

---

## IGNORE\_BASELINE= LIBNAME Statement Option

Specifies whether to try the connection regardless of the minimum required Hadoop version.

|              |                                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                               |
| Category:    | Data Set Control                                                                                                           |
| Default:     | NO                                                                                                                         |
| Data source: | Hadoop, Spark                                                                                                              |
| Notes:       | Support for this LIBNAME option was added in the spring release of SAS 9.4M5.<br>Support for Spark was added in SAS 9.4M7. |

## Syntax

**IGNORE\_BASELINE=YES | NO**

### Syntax Description

#### YES

Specifies that SAS allows connections to a target DBMS that does not meet minimum requirements. Although using this option is not recommended, it might be useful when version information is reported incorrectly or in test environments.

#### NO

Specifies that SAS rejects connections to a target DBMS that does not meet minimum requirements and issues an error to the SAS log.

## Details

Many SAS/ACCESS engines have a minimum supported version requirement. For details, see the configuration guide for the specific product. The IGNORE\_BASELINE= option checks the target DBMS version to see whether it meets the minimum baseline requirement. An error occurs if that condition is not met. For more information about Hadoop (Hive and Spark) minimum baseline requirements, see [Support for Hadoop 9.4](#).

Specifying IGNORE\_BASELINE=YES can be useful if, for example, you are testing a new version of SAS but have not yet upgraded your DBMS to the required version. Some DBMS systems might also report version information incorrectly. In that case, this option can be helpful.

### Example: Check Compatibility with an Existing Database

```
option set=SAS_HADOOP_JAR_PATH="c:\temp\mysrv1_jars"; /* site-specific */
option set=SAS_HADOOP_CONFIG_PATH="c:\temp\mysrv1_cfg"; /* site-specific */
options sastrace=',,,d' sastraceloc=saslog
      nostsuffix sql_ip_trace=(note,source) msglevel=i;
libname hdp hadoop user=myusr1 pw=mypwd1;
```

ERROR: The version of Hive on this cluster (0.13) does not meet the SAS minimum version requirements (1.1 or greater). Upgrade your version of Hive, set LIBNAME option IGNORE\_BASELINE=YES, or set the environment variable SAS\_ACCESS\_IGNORE\_BASELINE=YES. Very old versions of Hive might report version information incorrectly.

ERROR: Error in the LIBNAME statement.

```
libname hdp hadoop user=myusr1 pw=mypwd1 ignore_baseline=YES;
```

NOTE: Libref HDP was successfully assigned as follows:  
 Engine: HADOOP  
 Physical Name: jdbc:hive2:c:/mysrvm.it.com:10000/default

## IGNORE\_READ\_ONLY\_COLUMNS= LIBNAME Statement Option

Specifies whether to ignore or include columns where data types are read-only when generating an SQL statement for inserts or updates.

|              |                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                  |
| Category:    | Data Set Control                                                                                                                                                              |
| Alias:       | IGNORE_READONLY= [Greenplum, HAWQ, SAP IQ]                                                                                                                                    |
| Default:     | NO                                                                                                                                                                            |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, ODBC, OLE DB, PostgreSQL, SAP HANA, SAP IQ, Vertica, Yellowbrick |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">IGNORE_READ_ONLY_COLUMNS= data set option</a>                                                                                                                     |

## Syntax

**IGNORE\_READ\_ONLY\_COLUMNS=**[YES | NO](#)

### Syntax Description

#### YES

specifies that the SAS/ACCESS engine ignores columns where data types are read-only when you are generating insert and update SQL statements.

#### NO

specifies that the SAS/ACCESS engine does not ignore columns where data types are read-only when you are generating insert and update SQL statements.

## Details

Several databases include data types that can be read-only, such as the data type of the Microsoft SQL Server timestamp. Several databases also have properties that

allow certain data types to be read-only, such as the Microsoft SQL Server identity property.

When IGNORE\_READ\_ONLY\_COLUMNS=NO and a DBMS table contains a column that is read-only, an error is returned indicating that the data could not be modified for that column.

## Example

For this example, a database that contains the table Products is created with two columns: ID and PRODUCT\_NAME. The ID column is specified as a Read-only data type and PRODUCT\_NAME is a character column.

```
CREATE TABLE products (id int IDENTITY PRIMARY KEY, product_name varchar(40))
```

Assume you have a SAS data set that contains the name of your products, and you would like to insert the data into the Products table.

```
data work.products;
  id=1;
  product_name='screwdriver';
  output;
  id=2;
  product_name='hammer';
  output;
  id=3;
  product_name='saw';
  output;
  id=4;
  product_name='shovel';
  output;
run;
```

With IGNORE\_READ\_ONLY\_COLUMNS=NO (the default), an error is returned by the database because in this example the ID column cannot be updated. However, if you set the option to YES and execute a PROC APPEND, the append succeeds, and the SQL statement that is generated does not contain the ID column.

```
libname x odbc uid=myusr1 pwd=myusr1 dsn=lupinss
           ignore_read_only_columns=yes;
options sastrace=',,,d' sastraceloc=saslog nostsuffix;
proc append base=x.PRODUCTS data=work.products;
run;
```

## IMPALA\_PRINCIPAL= LIBNAME Statement Option

Specifies the Kerberos principal for the Impala server.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Access

Default: none

|               |                                                                                                                 |
|---------------|-----------------------------------------------------------------------------------------------------------------|
| Restrictions: | The IMPALA_PRINCIPAL= LIBNAME option applies only when you connect to the Impala server in a LIBNAME statement. |
|               | The IMPALA_PRINCIPAL= LIBNAME option applies only when you configure Impala to allow Kerberos authentication.   |
| Data source:  | Impala                                                                                                          |
| Note:         | Support for this LIBNAME option was added in SAS 9.4M2.                                                         |
| See:          | <a href="#">HDFS_PRINCIPAL= LIBNAME option</a> , “LIBNAME Statement for the Hadoop Engine”                      |

## Syntax

**IMPALA\_PRINCIPAL='principal'**

### Required Argument

#### *principal*

specifies the server’s Kerberos service principal name (SPN). Surround the principal value with single or double quotation marks. For example, you might specify a principal value that is similar to the following code:

```
impala_principal='impala/impala_host.example.com@TEST.EXAMPLE.COM'
```

## IN= LIBNAME Statement Option

Lets you specify the database and tablespace in which you want to create a new table.

|              |                                                              |
|--------------|--------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                 |
| Category:    | Data Set Control                                             |
| Alias:       | CREATED_IN= [DB2 under UNIX and PC Hosts]                    |
| Default:     | none                                                         |
| Data source: | DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                     |
| See:         | <a href="#">IN= data set option</a>                          |

## Syntax

**IN='database-name.tablespace-name' | database-name'**

### Syntax Description

#### *database-name.tablespace-name*

specifies the names of the database and tablespace, which are separated by a period. Enclose the entire specification in single quotation marks.

**DATABASE database-name**

specifies only the database name. Specify the word DATABASE, a space, and the database name. Enclose the entire specification in single quotation marks.

---

## Details

The IN= option is relevant only when you are creating a new table.

*DB2 under z/OS:* If you omit this option, the default is to create the table in the default database, implicitly creating a simple tablespace with a tablespace name that is generated by DB2.

---

## INSERT\_SQL= LIBNAME Statement Option

Determines the method to use to insert rows into a data source.

|              |                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                      |
| Category:    | Data Set Control                                                                                                                                                                  |
| Default:     | DBMS-specific                                                                                                                                                                     |
| Data source: | Amazon Redshift, Greenplum, HAWQ, Microsoft SQL Server, Netezza, ODBC, OLE DB, PostgreSQL, Vertica                                                                                |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in SAS Viya 3.4.</p> <p>Support for Netezza was added in SAS 9.4M7 and SAS Viya 3.5.</p> |
| See:         | <a href="#">INSERT_SQL= data set option</a> , <a href="#">INSERTBUFF= LIBNAME option</a> , <a href="#">INSERTBUFF= data set option</a>                                            |

---

## Syntax

**INSERT\_SQL=**[YES | NO](#)

### Syntax Description

#### **YES**

specifies that SAS/ACCESS uses the SQL insert method for the data source to insert new rows into a table.

#### **NO**

specifies that SAS/ACCESS uses an alternate, DBMS-specific method to insert new rows into a table.

## Details

Flat file databases such as dBASE, FoxPro, and text files generally have improved insert performance when INSERT\_SQL=NO. Other databases might have inferior insert performance or might fail with this value. You should therefore experiment to determine the optimal value to meet your needs.

*Greenplum, HAWQ, ODBC:* The default is YES, except for Microsoft Access, where the default is NO. When INSERT\_SQL=NO, the SQLSetPos (SQL\_ADD) function inserts rows in groups that are the size of the INSERTBUFF= option value. The SQLSetPos (SQL\_ADD) function does not work unless your driver supports it.

*Microsoft SQL Server:* The Microsoft SQL Server default is YES. When INSERT\_SQL=NO, the SQLSetPos (SQL\_ADD) function inserts rows in groups that are the size of the INSERTBUFF= option value. The SQLSetPos (SQL\_ADD) function does not work unless your driver supports it.

*Netezza, PostgreSQL, Vertica:* The default is YES.

*OLE DB:* By default, the OLE DB interface tries to use the most efficient row-insertion method for each data source. You can use the INSERT\_SQL option to override the default in the event that it is not optimal for your situation. Used when this option is set to NO, the alternate OLE DB method uses the OLE DB IRowsetChange interface.

---

## INSERTBUFF= LIBNAME Statement Option

Specifies the number of rows in a single DBMS insert.

|               |                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                         |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:      | DBMS-specific                                                                                                                                                                                                                                                                                                                                        |
| Restrictions: | <p>SAS allows the maximum number of rows that the DBMS allows, up to 32,767 rows.</p> <p>Additional driver-specific restrictions might apply.</p> <p>The optimal value for this option varies with factors such as network type and available memory.</p>                                                                                            |
| Interactions: | <p>If you specify the <b>DBCOMMIT=</b> option with a value that is less than the value of <b>INSERTBUFF=</b>, then <b>DBCOMMIT=</b> overrides <b>INSERTBUFF=</b>.</p> <p><i>DB2 under UNIX and PC Hosts, Greenplum:</i> To use this option, you must specify <b>INSERT_SQL=YES</b>.</p>                                                              |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Greenplum, HAWQ, Impala, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick                                                                                                                       |
| Notes:        | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |

- Tip: You might need to experiment with different values to determine the best value for your site.
- See: INSERTBUFF= data set option, DBCOMMIT= LIBNAME option, DBCOMMIT= data set option, INSERT\_SQL= LIBNAME option, INSERT\_SQL= data set option, READBUFF= LIBNAME option, READBUFF= data set option

## Syntax

**INSERTBUFF=***positive-integer*

### Syntax Description

***positive-integer***

specifies the number of rows to insert.

## Details

### Default Values for the INSERTBUFF= LIBNAME Option

Table 12.5 DBMS-Specific Default Values

| DBMS                                                                                                                         | Default                                                                                  |
|------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| Amazon Redshift                                                                                                              | 250                                                                                      |
| Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Greenplum, HAWQ, Impala, Netezza, PostgreSQL, SAP HANA, SAP IQ, Vertica | automatically calculated based on row length                                             |
| Microsoft SQL Server, MySQL, ODBC, OLE DB, Snowflake, Yellowbrick                                                            | 1                                                                                        |
| Oracle                                                                                                                       | When REREAD_EXPOSURE=YES, the (forced) default value is 1. Otherwise, the default is 10. |

### Items to Be Aware of for INSERTBUFF=

SAS application messages that indicate the success or failure of an Insert operation represent information for only a single insert, even when multiple inserts are

performed. Therefore, when you assign a value that is greater than INSERTBUFF=1, these messages might be incorrect.

When you insert rows with the VIEWTABLE window or the FSVIEW or FSEDIT procedure, use INSERTBUFF=1 to prevent the DBMS interface from trying to insert multiple rows. These features do not support inserting more than one row at a time.

*DB2 under UNIX and PC Hosts:* If one row in the insert buffer fails, all rows in the insert buffer fail.

*Impala:* If the calculated INSERTBUFF= value exceeds the default DBCOMMIT= value of 1000, the INSERTBUFF= value is likewise set to 1000. This helps to improve performance.

*MySQL:* Values greater than 0 activate the INSERTBUFF= option, and the engine calculates how many rows it can insert at one time, based on row size. If one row in the insert buffer fails, all rows in the insert buffer might fail, depending on your storage type.

---

## INTERFACE= LIBNAME Statement Option

Specifies the name and location of the interfaces file that is searched when you connect to the SAP ASE server.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: none

Data source: SAP ASE

---

## Syntax

**INTERFACE=<'>file-name<'>**

---

## Details

The interfaces file contains names and access information for the available servers on the network. If you omit a file name, the default action for your operating system occurs. INTERFACE= is not used in some operating environments. Contact your database administrator to see whether this statement applies to your computing environment.

---

## KEYSET\_SIZE= LIBNAME Statement Option

Specifies the number of rows that are keyset driven.

|              |                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement and some DBMS-specific connection options. See the DBMS-specific reference section for details. |
| Category:    | Data Set Control                                                                                                             |
| Alias:       | KEYSET= [Greenplum, HAWQ, Microsoft SQL Server]                                                                              |
| Default:     | 0                                                                                                                            |
| Interaction: | This option is valid only when CURSOR_TYPE=KEYSET_DRIVEN.                                                                    |
| Data source: | Amazon Redshift, Greenplum, HAWQ, Microsoft SQL Server, ODBC, PostgreSQL, Vertica                                            |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift and PostgreSQL was added in SAS Viya 3.4.            |
| See:         | <a href="#">CURSOR_TYPE= LIBNAME option</a> , <a href="#">KEYSET_SIZE= data set option</a>                                   |

---

## Syntax

**KEYSET\_SIZE=***number-of-rows*

### Syntax Description

***number-of-rows***

an integer with a value between 0 and the number of rows in the cursor.

---

## Details

If KEYSET\_SIZE=0, the entire cursor is keyset driven. If you specify a value greater than 0 for KEYSET\_SIZE=, that value indicates the number of rows within the cursor that functions as a keyset-driven cursor. When you scroll beyond the bounds that KEYSET\_SIZE= specifies, the cursor becomes dynamic and new rows might be included in the cursor. This becomes the new keyset, and the cursor functions as a keyset-driven cursor again. Whenever you specify a value between 1 and the number of rows in the cursor, the cursor is considered to be a mixed cursor: Part of it functions as a keyset-driven cursor and part functions as a dynamic cursor.

---

## LARGE\_RESULTS\_DATASET= LIBNAME Statement Option

Specifies a data set ID for query results.

|           |                              |
|-----------|------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement |
| Category: | Data Set Control             |
| Alias:    | LRD=                         |
| Default:  | _sasbq_temp_tables           |

|              |                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------|
| Requirement: | You must have Write permission to specify this option.                                                         |
| Interaction: | This option is used when the ALLOW_LARGE_RESULTS= LIBNAME option is enabled.                                   |
| Data source: | Google BigQuery                                                                                                |
| Note:        | Support for this option was added in the April 2021 update for SAS/ACCESS on SAS 9.4M7.                        |
| See:         | <a href="#">ALLOW_LARGE_RESULTS= LIBNAME option</a> ,<br><a href="#">LARGE_RESULTS_DATASET= LIBNAME option</a> |

## Syntax

**LARGE\_RESULTS\_DATASET=<'>*Google-BigQuery-data-set-ID*<'>**

### Required Argument

#### ***Google-BigQuery-data-set-ID***

specifies a data set ID to which result set tables are written.

## Details

If the specified data set ID does not exist, then it is created.

## LARGE\_RESULTS\_EXPIRATION\_TIME= LIBNAME Statement Option

Specifies the amount of time in milliseconds until a temporary table is deleted.

|              |                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                |
| Category:    | Data Set Control                                                                                            |
| Alias:       | LRET=                                                                                                       |
| Default:     | 86400000 (24 hours)                                                                                         |
| Range:       | 1 (see Details), 3600000–2147483647                                                                         |
| Interaction: | This option is used when the ALLOW_LARGE_RESULTS= LIBNAME option is enabled.                                |
| Data source: | Google BigQuery                                                                                             |
| Note:        | Support for this option was added in the April 2021 update for SAS/ACCESS on SAS 9.4M7.                     |
| See:         | <a href="#">ALLOW_LARGE_RESULTS= LIBNAME option</a> , <a href="#">LARGE_RESULTS_DATASET= LIBNAME option</a> |

## Syntax

**LARGE\_RESULTS\_EXPIRATION\_TIME=***milliseconds*

### Required Argument

***milliseconds***

specifies the amount of time in milliseconds before a temporary table is deleted.

---

## Details

If you specify the special value 1, the table does not expire.

---

## LOCATION= LIBNAME Statement Option

Allows further qualification of exactly where a table resides.

|              |                                                                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                |
| Category:    | Data Access                                                                                                                                                                                 |
| Alias:       | LOC=                                                                                                                                                                                        |
| Default:     | none                                                                                                                                                                                        |
| Requirement: | If you specify LOCATION=, you must also specify the AUTHID= LIBNAME option.                                                                                                                 |
| Data source: | DB2 under z/OS                                                                                                                                                                              |
| See:         | <a href="#">AUTHID= LIBNAME option</a> , <a href="#">LOCATION= data set option</a> , <a href="#">REMOTE_DBTYPE= LIBNAME option</a> [to access a database server on Linux, UNIX, or Windows] |

---

## Syntax

**LOCATION=***location*

---

## Details

The location name maps to the location in the SYSIBM.LOCATION catalog in the communication database.

In SAS/ACCESS Interface to DB2 under z/OS, the location is converted to the first level of a three-level table name: *location.authid.table*. The DB2 Distributed Data Facility (DDF) makes the connection implicitly to the remote DB2 subsystem when DB2 receives a three-level name in an SQL statement.

If you omit this option, SAS accesses the data from the local DB2 database unless you have specified a value for the SERVER= option. This option is not validated until you access a DB2 table.

---

## LOCKTABLE= LIBNAME Statement Option

Places exclusive or shared locks on tables.

|              |                                            |
|--------------|--------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement               |
| Category:    | Data Access                                |
| Default:     | no locking                                 |
| Data source: | Informix                                   |
| See:         | <a href="#">LOCKTABLE= data set option</a> |

---

## Syntax

**LOCKTABLE=EXCLUSIVE | SHARE**

### Syntax Description

#### **EXCLUSIVE**

specifies that other users are prevented from accessing each table that you open in the libref.

#### **SHARE**

specifies that other users or processes can read data from the tables, but they cannot update the data.

---

## Details

You can lock tables only if you are the owner or have been granted the necessary privilege.

---

## LOCKTIME= LIBNAME Statement Option

Specifies the number of seconds to wait until rows are available for locking.

|              |                                                                |
|--------------|----------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                   |
| Category:    | Data Access                                                    |
| Default:     | none                                                           |
| Requirement: | You must specify LOCKWAIT=YES for LOCKTIME= to have an effect. |

Data source: Informix  
 See: [LOCKWAIT= LIBNAME option](#)

## Syntax

**LOCKTIME=***positive-integer*

## Details

If you omit the LOCKTIME= option and use [LOCKWAIT=YES](#), SAS suspends your process indefinitely until a lock can be obtained.

## LOCKWAIT= LIBNAME Statement Option

Specifies whether to wait indefinitely until rows are available for locking.

Valid in: SAS/ACCESS LIBNAME statement  
 Category: Data Access  
 Default: DBMS-specific  
 Data source: Informix, Oracle  
 See: [LOCKTIME= LIBNAME option](#)

## Syntax

**LOCKWAIT=**YES | NO

## Syntax Description

### YES

specifies that SAS waits until rows are available for locking.

### NO

specifies that SAS does not wait and returns an error to indicate that the lock is not available.

## LOGDB= LIBNAME Statement Option

Redirects to an alternate database-specific table that FastExport creates or MultiLoad uses.

Valid in: DATA and PROC steps, wherever you use FastExport or MultiLoad

|              |                                                                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Category:    | Data Set Control                                                                                                                                                                         |
| Default:     | default Teradata database for the libref                                                                                                                                                 |
| Data source: | Teradata                                                                                                                                                                                 |
| Tip:         | You can also use LOGDB= with TPT options.                                                                                                                                                |
| See:         | <a href="#">FASTEXPORT= LIBNAME option</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">TPT= LIBNAME option</a> , <a href="#">TPT= data set option</a> , “Using MultiLoad” |

## Syntax

**LOGDB=*database-name***

### Syntax Description

***database-name***

specifies the name of the Teradata database.

## Details

[Teradata Fast Export utility](#): The FastExport restart capability is not yet supported. When you use this option with FastExport, FastExport creates restart log tables in an alternate database. You must have the necessary permissions to create tables in the specified database, and FastExport creates only restart tables in that database.

[Teradata MultiLoad utility](#) : To specify this option, you must first specify **MULTILOAD=YES**. When you use this option with the Teradata MultiLoad utility, MultiLoad redirects the restart table, the work table, and the required error tables to an alternate database.

## Examples

### Example 1: Create Restart Log Tables

In this example, PROC PRINT calls the Teradata FastExport utility, if it is installed. FastExport creates restart log tables in the ALTDB database.

```
libname mydblib teradata user=myusr1 pw=mypwd1 logdb=altdb;
proc print data=mydblib.mytable(dbsliceparm=all);
run;
```

### Example 2: Create Restart, Work, and Error Tables

In this next example, MultiLoad creates the restart table, work table, and error tables in the alternate database that LOGDB= specifies.

```
/* Create work tables in altdb2 database,
```

```

where I have create & drop privileges. */
libname x teradata user=myusr1 pw=xxxxx logdb=altdb2;
data x.testload(multiload=YES);
do i=1 to 100;
output;
end;
run;

```

## Example 3: Create the Work Table in a Different Database

Using MultiLoad with the TPT API, this example provides a different name for the work table and redirects the table to the AUDATA00\_work database.

```

libname tera teradata user=myusr1 pw=mypwd1 logdb=audata00_work;
data teratestdata(MULTILOAD=YES TPT_WORK_TABLE=work);
i=1;output; i=2;output;
run;

```

## LOGIN\_TIMEOUT= LIBNAME Statement Option

Specifies the default login time-out for connecting to and accessing data sources in a library.

|              |                                                                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                                |
| Category:    | Data Access                                                                                                                                                                                                                                                    |
| Defaults:    | 30 [Hadoop, Spark]<br>0 [Amazon Redshift, Aster, Impala, JDBC, Microsoft SQL Server, Netezza, ODBC, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Yellowbrick]                                                                                                      |
| Data source: | Amazon Redshift, Aster, Hadoop, Impala, JDBC, Microsoft SQL Server, Netezza, ODBC, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Spark, Yellowbrick                                                                                                                 |
| Notes:       | Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for JDBC, Microsoft SQL Server, and PostgreSQL was added in SAS Viya 3.4.<br>Support for Spark was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |

## Syntax

**LOGIN\_TIMEOUT=***numeric-value*

### Syntax Description

#### ***numeric-value***

specifies a positive integer for the number of seconds to wait for the connection. A value of 0 indicates to wait indefinitely.

---

## MAX\_BINARY\_LEN= LIBNAME Statement Option

Specifies the maximum length in bytes of a binary column.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: 2000

Data source: Google BigQuery

Note: Support for this option was added in the August 2019 release of SAS/ACCESS.

---

### Syntax

**MAX\_BINARY\_LEN=***value*

### Syntax Description

***value***

specifies the maximum length in bytes of a binary column. The value that you specify must be greater than 0.

---

## MAX\_CHAR\_LEN= LIBNAME Statement Option

Specifies the maximum number of characters to allocate for Google BigQuery string columns.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: 2000

Restriction: See the Details section for information about when this option is honored.

Data source: Google BigQuery

Note: Support for this option was added in the August 2019 release of SAS/ACCESS.

---

### Syntax

**MAX\_CHAR\_LEN=***value*

## Syntax Description

### **value**

specifies the maximum number of characters to allocate for string columns.

---

## Details

Use this option when you work with string types in Google BigQuery. By default, in Google BigQuery strings do not have a fixed length. However, SAS requires a specified length for these values. Specify a length that is long enough to contain all of the string data that you are working with to avoid truncation of string values.

This option pertains to Read operations but not to Write operations for new tables. Write operations on existing tables are honored by this option.

---

## MAX\_CONNECTS= LIBNAME Statement Option

Specifies the maximum number of simultaneous connections that SAP ASE allows.

|              |                              |
|--------------|------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement |
| Category:    | Data Access                  |
| Default:     | 25                           |
| Data source: | SAP ASE                      |

---

## Syntax

**MAX\_CONNECTS=***numeric-value*

---

## Details

If you omit MAX\_CONNECTS=, the default for the maximum number of connections is 25. Increasing the number of connections has a direct impact on memory.

---

## MAX\_STRING\_SIZE= LIBNAME Statement Option

Specifies how to write character values to tables in Oracle.

|           |                              |
|-----------|------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement |
| Category: | Data Set Control             |
| Default:  | NONE                         |

Data source: Oracle

Note: Support for this option was added in SAS 9.4M8.

## Syntax

**MAX\_STRING\_SIZE=NONE | EXTENDED | STANDARD**

### Required Arguments

#### **NONE**

specifies that SAS/ACCESS uses the MAX\_STRING\_SIZE value from the Oracle configuration file.

#### **EXTENDED**

specifies that SAS/ACCESS writes character values to an Oracle VARCHAR2 variable that can contain up to 32767 bytes.

#### **STANDARD**

specifies that SAS/ACCESS writes character values that are 4000 bytes or less to an Oracle VARCHAR2 variable. Character values that are longer than 4000 bytes are written to CLOB variables.

## Details

This option is used when creating tables in Oracle that are populated from SAS data sets. If a text variable is 32767 bytes or less and if MAX\_STRING\_SIZE=EXTENDED, then that variable is written to Oracle as a VARCHAR2. However, if a text value is more than 4000 bytes in length and MAX\_STRING\_SIZE=STANDARD, then the variable is created as a CLOB in Oracle. If MAX\_STRING\_SIZE=STANDARD and a text value is 4000 bytes or less, then the value is written to Oracle as a VARCHAR2.

The value that you specify for this option should match the value of the MAX\_STRING\_SIZE parameter in the Oracle configuration file. In some cases, Oracle returns an incorrect value for the MAX\_STRING\_SIZE parameter. In those cases, use this option to explicitly set the MAX\_STRING\_SIZE= LIBNAME option to the value of the MAX\_STRING\_SIZE parameter in Oracle.

## MODE= LIBNAME Statement Option

Specifies whether the connection to Teradata uses the ANSI or Teradata mode.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: ANSI

Data source: Teradata

See: [SQL Pass-Through Facility Specifics for Teradata](#)

## Syntax

**MODE=TERADATA | ANSI**

### Syntax Description

#### TERADATA

specifies that SAS/ACCESS opens Teradata connections in Teradata mode.

#### ANSI

specifies that SAS/ACCESS opens Teradata connections in ANSI mode.

## Details

This option allows Teradata connections to open in the specified mode. Connections that open with MODE=TERADATA use Teradata mode rules for all SQL requests that are passed to the Teradata DBMS. This impacts transaction behavior and can cause case insensitivity when processing data.

During data insertion, not only is each inserted row committed implicitly, but rollback is not possible when the error limit is reached if you also specify **ERRLIMIT=**. Any update or deletion that involves a cursor does not work.

ANSI mode is recommended for all features that SAS/ACCESS supports, and Teradata mode is recommended only for reading data from Teradata.

## Example

This example does not work because it requires the use of a cursor.

```
libname x teradata user=myusr1 pw=XXXX mode=teradata;
/* Fails with "ERROR: Cursor processing is
not allowed in Teradata mode." */
proc sql;
update x.test
set i=2;
quit;
```

This example works because the DBIDIRECTEXEC= system option sends the delete SQL directly to the database without using a cursor.

```
libname B teradata user=myusr1 pw=XXXX mode=Teradata;
options dbidirectexec;
proc sql;
delete from b.test where i=2;
quit;
```

---

# MULTI\_DATASRC\_OPT= LIBNAME Statement Option

Used in place of DBKEY to improve performance when processing a join between two data sources.

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                         |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:     | NONE                                                                                                                                                                                                                                                                                                                                                 |
| Restriction: | This option is used only for PROC SQL processing. It is not used in DATA step processing.                                                                                                                                                                                                                                                            |
| Interaction: | This option is not used when a value is specified for the DBKEY= data set option.                                                                                                                                                                                                                                                                    |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                  |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">DBKEY= data set option</a> , <a href="#">DBLARGETABLE= data set option</a>                                                                                                                                                                                                                                                               |

---

## Syntax

**MULTI\_DATASRC\_OPT=IN\_CLAUSE | NONE**

### Syntax Description

#### **IN\_CLAUSE**

specifies use of an IN clause that contains values that were read from a smaller table. The clause is used to retrieve matching values in a larger table based on a key column that was designated in an equijoin.

#### **NONE**

turns off option functionality.

---

## Details

When you are processing a join between a SAS data set and a DBMS table, the SAS data set should be smaller than the DBMS table for optimal performance.

However, if the SAS data set is larger than the DBMS table, the SAS data set is still used in the IN clause.

When SAS processes a join between two DBMS tables, SELECT COUNT (\*) is issued to determine which table is smaller and whether it qualifies for an IN clause. You can use the DBLARGETABLE= data set option to prevent the SELECT COUNT (\*) from being issued.

The IN clause currently has a limit of 4,500 unique values.

Specifying **DBKEY=** automatically overrides **MULTI\_DATASRC\_OPT=**.

**DIRECT\_SQL=** can affect this option as well. If **DIRECT\_SQL=NONE** or **NOWHERE**, the IN clause cannot be built and passed to the DBMS, regardless of the value of **MULTI\_DATASRC\_OPT=**. These values for **DIRECT\_SQL=** prevent a WHERE clause from being passed.

**Oracle:** Oracle can handle an IN clause of only 1,000 values. It therefore divides larger IN clauses into multiple smaller IN clauses. The results are combined into a single result set. For example, if an IN clause contained 4,000 values, Oracle produces 4 IN clauses that contain 1,000 values each. A single result is produced, as if all 4,000 values were processed as a whole.

**OLE DB:** OLE DB restricts the number of values allowed in an IN clause to 255.

## Examples

### Example 1: Build and Pass an IN Clause for a Join

This example builds and passes an IN clause from the SAS table to the DBMS table, retrieving only the necessary data to process the join.

```
proc sql;
create view work.v as
select tab2.deptno, tab2.dname from
work.sastable tab1, dblib.table2 tab2
where tab12.deptno = tab2.deptno
using libname dblib oracle user=myusr1 password=mypwd1
  multi_datasrc_opt=in_clause;
quit;
```

### Example 2: Prevent Build and Pass of an IN Clause for a Join

This example prevents the building and passing of the IN clause to the DBMS. It requires all rows from the DBMS table to be brought into SAS to process the join.

```
libname dblib oracle user=myusr1 password=mypwd1 multi_datasrc_opt=none;
proc sql;
  select tab2.deptno, tab2.dname from
    work.table1 tab1,
    dblib.table2 tab2
  where tab1.deptno=tab2.deptno;
quit;
```

---

## MULTISTMT= LIBNAME Statement Option

Specifies whether insert statements are sent to Teradata one at a time or in a group.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: NO

Restriction: You currently cannot use MULTISTMT= with ERRLIMIT=.

Data source: Teradata

See: [MULTISTMT= data set option](#), [QUERY\\_BAND= data set option](#), [TPT\\_APPL\\_PHASE= data set option](#), [TPT\\_BUFFER\\_SIZE= data set option](#), [TPT\\_DATA\\_ENCRYPTION= data set option](#), [TPT\\_DATA\\_ENCRYPTION= LIBNAME option](#), [TPT\\_ERROR\\_TABLE\\_1= data set option](#), [TPT\\_ERROR\\_TABLE\\_2= data set option](#), [TPT\\_LOG\\_TABLE= data set option](#), [TPT\\_MAX\\_SESSIONS= LIBNAME option](#), [TPT\\_MAX\\_SESSIONS= data set option](#), [TPT\\_MIN\\_SESSIONS= data set option](#), [TPT\\_PACK= data set option](#), [TPT\\_PACKMAXIMUM= data set option](#), [TPT\\_RESTART= data set option](#), [TPT\\_TRACE\\_LEVEL= data set option](#), [TPT\\_TRACE\\_LEVEL\\_INF= data set option](#), [TPT\\_TRACE\\_OUTPUT= data set option](#), [TPT\\_UNICODE\\_PASSTHRU= LIBNAME option](#)

---

## Syntax

**MULTISTMT=YES | NO**

### Syntax Description

#### YES

use TPT stream processing and use the maximum buffer size available for the Teradata Client TTU version and the Teradata DBS version that is being accessed.

#### NO

send inserts to Teradata one row at a time.

---

## Details

When you request multistatement inserts, SAS first determines how many insert statements it can send to Teradata. Several factors determine the actual number of statements that SAS can send—for example, how many:

- SQL insert statements can fit in the available buffer.
- data rows can fit in the available buffer.
- inserts the Teradata server chooses to accept.

When you need to insert large volumes of data, you can significantly improve performance by using MULTISTMT= instead of inserting only single-row.

If you also specify [DBCOMMIT=](#), SAS uses the smaller of these: the DBCOMMIT= value and the number of insert statements that can fit in a buffer as the number of insert statements to send together at one time.

---

## OR\_BINARY\_DOUBLE= LIBNAME Statement Option

Specifies the default data type to use for numeric table columns.

|              |                                                               |
|--------------|---------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                  |
| Category:    | Data Set Control                                              |
| Default:     | YES                                                           |
| Requirement: | Oracle Server 10g or higher                                   |
| Data source: | Oracle                                                        |
| Note:        | In SAS 9.4M1, the default for this LIBNAME option became YES. |
| See:         | <a href="#">DBTYPE= data set option</a>                       |

---

## Syntax

**OR\_BINARY\_DOUBLE =**[YES | NO](#)

### Syntax Description

**YES**

specifies BINARY\_DOUBLE as the default.

**NO**

specifies NUMBER as the default.

---

## Details

Use this option when you want a wider range of numbers than when compared to the NUMBER. You can override this option with the [DBTYPE=data set option](#).

---

## OR\_ENABLE\_INTERRUPT= LIBNAME Statement Option

Allows interruption of any long-running SQL processes on the DBMS server.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: NO

Data source: Oracle

---

### Syntax

**OR\_ENABLE\_INTERRUPT=YES | NO**

#### Syntax Description

##### **YES**

allows interruption of long-running SQL processes on the DBMS server.

##### **NO**

disables interruption of long-running SQL processes on the DBMS server.

---

### Details

OR\_ENABLE\_INTERRUPT enables you to interrupt long-running queries that you submitted through the SAS interface. You can use this option to interrupt these statements:

- any SELECT SQL statement that was submitted by using the *SELECT \* FROM CONNECTION* as a pass-through statement
- any statement other than the SELECT SQL statement that you submitted by using the EXECUTE statement as a pass-through statement

---

## OR\_UPD\_NOWHERE= LIBNAME Statement Option

Specifies whether SAS uses an extra WHERE clause when updating rows with no locking.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

|              |                                                                                                                                                                                                                                 |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alias:       | ORACLE_73_OR_ABOVE=                                                                                                                                                                                                             |
| Default:     | YES                                                                                                                                                                                                                             |
| Requirement: | Due to the published Oracle bug 440366, an update on a row sometimes fails even if the row has not changed. Oracle offers this solution: When you create a table, increase the number of INITTRANS to at least 3 for the table. |
| Data source: | Oracle                                                                                                                                                                                                                          |
| See:         | <a href="#">"Locking in the Oracle Interface"</a> , OR_UPD_NOWHERE= data set option,<br><a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a>                                                                                        |

## Syntax

**OR\_UPD\_NOWHERE=YES | NO**

### Syntax Description

#### YES

specifies that SAS does not use an additional WHERE clause to determine whether each row has changed since it was read. Instead, SAS uses the SERIALIZABLE isolation level (available with Oracle7.3 and above) for update locking. If a row changes after the serializable transaction starts, the update on that row fails.

#### NO

specifies that SAS uses an additional WHERE clause to determine whether each row has changed since it was read. If a row has changed since being read, the update fails.

## Details

Use this option when you are updating rows without locking ([UPDATE\\_LOCK\\_TYPE=NOLOCK](#)).

By default (OR\_UPD\_NOWHERE=YES), updates are performed in serializable transactions. It lets you avoid extra WHERE-clause processing and potential WHERE-clause floating-point precision problems.

## OVERRIDE\_RESP\_LEN= LIBNAME Statement Option

Specifies whether to override the default response buffer length.

|           |                              |
|-----------|------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement |
| Category: | Data Set Control             |
| Default:  | NO                           |

|              |                                                                                                                                                                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Interaction: | When you set this option to YES, specify a response buffer length larger than the default value (1,048,500 bytes) in the Teradata configuration file (clispb.dat). Also, specify the file name and location of the configuration file in the COPLIB environment variable. |
| Data source: | Teradata                                                                                                                                                                                                                                                                  |
| Note:        | Support for this option was added in SAS 9.4M8.                                                                                                                                                                                                                           |
| Tip:         | Use this option to improve performance when you are not using the Teradata TPT API.                                                                                                                                                                                       |
| Example:     | <pre>options set=COPLIB="/u/user/config/4MBbuff"; libname td teradata server=myserver user=myuser password=mypwd override_resp_len=yes;  data null;   set td.table-name (tpt=no);   &lt;other statements&gt; run;</pre>                                                   |

## Syntax

**OVERRIDE\_RESP\_LEN=YES | NO**

### Required Argument

**YES | NO**

specifies whether to override the default response buffer length.

## Details

Use this option to enhance performance when you are not using the Teradata TPT API.

The default response buffer length that is used by SAS/ACCESS is 1,048,500 bytes. Specify a different buffer length for the resp\_buf\_len variable in the Teradata configuration file (clispb.dat). Also, specify that the COPLIB environment variable points to the configuration file with the modified buffer length. The path that you supply for COPLIB should contain the location of the configuration file. The configuration file name is assumed to be clispb.dat.

Your modified configuration file might contain this line to set the response buffer size to 4 MB.

```
resp_buf_len=4194304
```

You can specify the OVERRIDE\_RESP\_LEN= option in the CONNECT statement in a call to PROC SQL. For example, you might specify code similar to this sample. SAS/ACCESS uses the response buffer length from the clispb.dat file in /u/myuser/project/4MBbuff.

```
options set=COPLIB="/u/myuser/project/4MBbuff";

proc sql;
connect to teradata (database=mydatabase tdpid=value user=myUID
password=mypwd1
```

```

        tpt=no override_resp_len=yes);
create table largerow as select * from connection to
teradata
(select * from myUID.SampleLargeRow);

disconnect from teradata;
quit;

```

---

## PACKETSIZE= LIBNAME Statement Option

Allows specification of the packet size for SAP ASE to use.

Valid in: SAS/ACCESS LIBNAME statement  
 Category: Data Set Control  
 Default: current server setting  
 Data source: SAP ASE

---

## Syntax

**PACKETSIZE=***numeric-value*

### Syntax Description

***numeric-value***

any multiple of 512, up to the limit of the maximum network packet size value on your server.

---

## Details

If you omit PACKETSIZE=, the default is the current server value. You can query the default network packet value in the ISQL utility by using the SAP ASE `sp_configure` command.

---

## PARMDEFAULT= LIBNAME Statement Option

Specifies whether the SAP HANA engine uses the defaults for variables and parameters that are specified in the metadata in SAP HANA.

Valid in: SAS/ACCESS LIBNAME statement  
 Category: Data Set Control  
 Alias: USE\_PARAMETER\_DEFAULT  
 Default: none

Data source: SAP HANA

## Syntax

**PARMDEFAULT=YES | NO**

### Syntax Description

#### YES

specifies to look up metadata for variables and parameters. This enables you to apply defaults for variables in a WHERE clause, and defaults for input parameters using the PLACEHOLDER syntax.

#### NO

specifies to not look up metadata for variables and parameters.

## Details

The default for the PARMDEFAULT= option is YES if the PARMSTRING=LIBNAME= or data set option is specified. The default is NO if no PARMSTRING= option is specified.

Applying the defaults requires additional queries to the metadata. It can be switched off to avoid making unnecessary queries.

## PARMSTRING= LIBNAME Statement Option

Specifies a quoted string of variable name and value pairs separated by a comma, or a placeholder string.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Alias: PARAMETERS

Default: none

Data source: SAP HANA

## Syntax

```
PARMSTRING=<"variable-name1=variable-value1,variable-name2=variable-value2,  
...>  
< "PLACEHOLDER' = ('variable-name1', 'variable-value1'),< 'PLACEHOLDER'=  
(variable-name2', 'variable-value2'),>...">
```

## Syntax Description

**"variable-name1=variable-value1"**

specifies the variable name and value pair. More than one pair can be specified and must be separated by a comma.

**"'PLACEHOLDER' = ('variable-name1', 'variable-value1')"**

specifies the variable name and value pair as a placeholder.

---

**Note:** You can also combine a name and value pair and a placeholder:

```
PARMSTRING="variable-name1=variable-value1",
<'PLACEHOLDER'=('variable-name2', 'variable-value2')"
```

---

## Details

When you specify the variable name and value pairs, the SAS/ACCESS engine locates the variable input parameter in the SAP HANA metadata. The value is applied either as a WHERE clause for variables, or it generates and applies a PLACEHOLDER= string for passing the input parameters to the view execution. If the variable input parameter is not found in metadata, it is appended as part of a PLACEHOLDER= string to the generated SQL string.

If the user specifies a placeholder string, the string is passed directly to the SAP HANA query to be processed in SAP HANA.

Here are some syntax examples:

```
PARMSTRING = "parm_price=30"

PARMSTRING = "'PLACEHOLDER' = ('$$parm_product$$', 'Tablet')"

PARMSTRING = "PLACEHOLDER.'$$parm_category$$'=>'Notebooks'"
```

When a PARMSTRING= LIBNAME option and a PARMSTRING= data set option are both specified, the PLACEHOLDER= string becomes a combination of both of the parameters and is passed to SAP HANA. An input parameter can occur only once as a placeholder in the SQL statement. If a parameter appears in the LIBNAME option and the data set option, the SAS/ACCESS engine tries to resolve this by passing only a fragment from the data set option.

---

## Comparisons

The PARMSTRING= LIBNAME option is applied to all column tables and column views in the library. It is not applied to row store objects.

The PARMSTRING= data set option is applied to the table or view that is specified.

---

## PARTITION\_KEY= LIBNAME Statement Option

Specifies the column name to use as the partition key for creating fact tables.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: none

Requirements: To create a data set in Aster without error, either specify DIMENSION=YES for the LIBNAME or data set option or specify a partition key in the PARTITION\_KEY= LIBNAME or data set option.

You must enclose the column name in quotation marks.

Data source: Aster

See: [DIMENSION= LIBNAME option](#), [DIMENSION= data set option](#), [PARTITION\\_KEY= data set option](#)

---

## Syntax

**PARTITION\_KEY='column-name'**

---

## Details

Aster uses dimension and fact tables.

---

## Example: Create a Dimension Table

This first example shows how you can use the SAS data set, SASFLT.flightschedule, to create an Aster dimension table, flightschedule, by using the DIMENSION= DATA step option.

```
LIBNAME sasflt 'SAS-library';
LIBNAME net_air ASTER user=myusr1 pwd=mypwd1 server=air2 database=flights;
data net_air.flightschedule(dimension=yes);
  set sasflt. flightschedule;
run;
```

You can create the same Aster dimension table by specifying [DIMENSION=YES](#) in the LIBNAME statement.

```
LIBNAME sasflt 'SAS-library';
LIBNAME net_air ASTER user=myusr1 pwd=mypwd1 server=air2
  database=flights dimension=yes;
data net_air.flightschedule;
  set sasflt. flightschedule;
```

```
run;
```

If you do not specify DIMENSION=YES by using either the LIBNAME or data set option, the Aster engine tries to create an Aster fact table. To do this, however, you must specify the PARTITION\_KEY= LIBNAME or data set option, as shown in this example.

```
LIBNAME sasflt 'SAS-library';
LIBNAME net_air ASTER user=myusr1 pwd=mypwd1 server=air2 database=flights;
data net_air.flightschedule(dbtype=(flightnumber=integer)
    partition_key='flightnumber');
    set sasflt. flightschedule;
run;
```

You can create the same Aster fact table by using the PARTITION\_KEY= LIBNAME option.

```
LIBNAME sasflt 'SAS-library';
LIBNAME net_air ASTER user=myusr1 pwd=mypwd1 server=air2 database=flights
    partition_key='flightnumber';
data net_air.flightschedule(dbtype=(flightnumber=integer));
    set sasflt. flightschedule;
run;
```

The above examples use the DBTYPE= data set option so that the data type of the partition-key column meets the limitations of the Aster partition-key column.

## POST\_DML\_STMT\_OPTS= LIBNAME Statement Option

Specifies text to append to an INSERT, UPDATE, or DELETE statement.

|              |                                                     |
|--------------|-----------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                        |
| Category:    | Data Access                                         |
| Default:     | none                                                |
| Data source: | Oracle                                              |
| Note:        | Support for this option was added in SAS 9.4M5.     |
| See:         | <a href="#">POST_DML_STMT_OPTS= data set option</a> |

## Syntax

**POST\_DML\_STMT\_OPTS=*value***

### Syntax Description

#### *value*

specifies text to append to an UPDATE, INSERT, or DELETE statement.  
SAS/ACCESS does not verify the validity of the text that you provide.

## Details

Typically, you use POST\_DML\_STMT\_OPTS= for logging errors that might occur during data manipulation (DML). In this case, you must have created an Oracle log table either outside SAS or by using explicit SQL code with PROC SQL. The text that you provide for the POST\_DML\_STMT\_OPTS= LIBNAME option should take this form (information within '< >' is optional):

```
LOG ERRORS <INTO <schema.>table-name> <('simple-expression')>
      <REJECT LIMIT integer | UNLIMITED>
```

This option acts only on tables that immediately follow 'INSERT INTO', 'UPDATE', or 'DELETE FROM' in your PROC SQL queries.

To see the SQL code that is generated, use the SASTRACE and SASTRACELOG system options, as shown here:

```
option sastraceloc=saslog sastrace=',,d';
```

Here is an example that uses the POST\_DML\_STMT\_OPTS= LIBNAME option to log an unlimited number of errors into the log table ERR\_DML. In this case, the errors logged are those that occur while inserting records into the Oracle table ORATAB.

```
LIBNAME oralib1 ORACLE PATH=orclpdb USER=orauser PASSWORD=xxxx
      POST_DML_STMT_OPTS=('LOG ERRORS INTO ERR_DML REJECT LIMIT UNLIMITED');

proc sql;
  insert into oralib1.ORATAB
  select *
  from oralib1.CLASS;
quit;
```

The example above generates the following SQL code:

```
insert into ORATAB ("NAME", "SEX", "AGE", "HEIGHT", "WEIGHT")
select TXT_2."NAME", TXT_2."SEX", TXT_2."AGE", TXT_2."HEIGHT", TXT_2."WEIGHT"
from CLASS TXT_2 LOG ERRORS INTO ERR_DML REJECT LIMIT UNLIMITED
```

---

## POST\_STMT\_OPTS= LIBNAME Statement Option

Allows additional database-specific options to be placed after the CREATE TABLE statement in generated SQL code.

|              |                                                                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                           |
| Category:    | Data Set Control                                                                                                                                                                                                       |
| Alias:       | DBCREATE_TABLE_OPTS=                                                                                                                                                                                                   |
| Default:     | none                                                                                                                                                                                                                   |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, Netezza, Oracle, PostgreSQL, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica |
| Notes:       | Support for this option was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.                                                                                      |

Support for JDBC was added in SAS Viya 3.4.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

See: [DBCREATE\\_TABLE\\_OPTS= LIBNAME option](#), [DBDIRECTEXEC system option](#), [POST\\_STMT\\_OPTS= data set option](#)

## Syntax

**POST\_STMT\_OPTS='DBMS-SQL-options'**

### Required Argument

#### DBMS-SQL-option(s)

specifies database-specific options to be placed after the CREATE TABLE statement. Enclose the options that you specify within single or double quotation marks.

## Details

You can use POST\_STMT\_OPTS= to add DBMS-specific clauses to the end of the SQL CREATE TABLE statement. The SAS/ACCESS engine passes the SQL CREATE TABLE statement and its clauses to the DBMS, which executes the statement and creates the DBMS table. POST\_STMT\_OPTS= applies only when you are creating a DBMS table by specifying a libref that is associated with DBMS data.

## PRESERVE\_COL\_NAMES= LIBNAME Statement Option

Preserves spaces, special characters, and case sensitivity in DBMS column names when you create DBMS tables.

|               |                                                                                                                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement (when you create DBMS tables)                                                                                                                                                                                                                             |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                       |
| Aliases:      | DBMIXED= [Impala]<br>PRES_COL= [Impala]<br>PRESERVE_NAMES= (see “Details”)<br>QUOTE_NAMES= [Impala, SAP IQ]                                                                                                                                                                            |
| Defaults:     | YES [Greenplum, HAWQ, Impala, MySQL, ODBC to Microsoft SQL Server, Snowflake, Teradata, Yellowbrick]<br>NO [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Hadoop, Informix, JDBC, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Spark, Vertical] |
| Restrictions: | This option applies only when you use SAS/ACCESS to create a new DBMS table.                                                                                                                                                                                                           |

|              |                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | PRESERVE_COL_NAMES= does not apply to the SQL pass-through facility.                                                                                                                                                                                                                                                                                                              |
| Interaction: | If you use the DS2 or FedSQL language, quoting and casing of names is different. For more information, see the identifiers topic in <a href="#">SAS DS2 Language Reference</a> or <a href="#">SAS FedSQL Language Reference</a> .                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                                                  |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Spark was added in SAS 9.4M7.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">PRESERVE_COL_NAMES= data set option</a> , <a href="#">SAS Names and Support for DBMS Names</a> , <a href="#">VALIDVARNAME= system option</a>                                                                                                                                                                                                                          |

---

## Syntax

**PRESERVE\_COL\_NAMES=YES | NO**

### Syntax Description

#### YES

specifies that column names that are used in table creation are passed to the DBMS with special characters and the exact, case-sensitive spelling of the name is preserved.

#### NO

specifies that column names that are used to create DBMS tables are derived from SAS variable names (VALIDVARNAME= system option) by using the SAS variable name normalization rules. However, the database applies its DBMS-specific normalization rules to the SAS variable names when creating the DBMS column names.

The use of N-literals to create column names that use database keywords or special symbols other than the underscore character might be invalid when DBMS normalization rules are applied. To include nonstandard SAS symbols or database keywords, specify PRESERVE\_COL NAMES=YES.

NO is the default for most DBMS interfaces.

---

## Details

When you create a table, you assign the column names by using one of these methods.

- To control the case of the DBMS column names, specify variables using the case that you want and specify PRESERVE\_COL NAMES=YES. If you use special

symbols or blanks, you must set VALIDVARNAME= to ANY and use N-literals. For more information, see the SAS/ACCESS naming topic in the DBMS-specific reference section for your interface in this document and also [SAS Data Set Options: Reference](#).

**SAP HANA:** When you specify PRESERVE\_COL\_NAMES=YES, you can use reserved words for column names.

- To enable the DBMS to normalize the column names according to its naming conventions, specify variables using any case and set PRESERVE\_COL\_NAMES= NO.

When you use SAS/ACCESS to read from, insert rows into, or modify data in an existing DBMS table, SAS identifies the database column names by their spelling. Therefore, when the database column exists, the case of the variable does not matter.

**Amazon Redshift, Hadoop, Spark:** The SAS/ACCESS engine automatically converts all schema, table, and column names to lowercase.

To save some time when coding, specify the PRESERVE\_NAMES= alias if you plan to specify both the PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options in your LIBNAME statement.

To use column names that are not valid SAS names in your SAS program, you must use one of these techniques.

- Use the DQUOTE= option in PROC SQL and reference your columns using double quotation marks. Here is an example.

```
proc sql dquote=ansi;
    select "Total$Cost" from mydblib.mytable;
```

- Specify the global system option VALIDVARNAME=ANY and use name literals in the SAS language. Here is an example.

```
proc print data=mydblib.mytable;
    format 'Total$Cost'n 22.2;
```

If you are creating a table in PROC SQL, you must also include the PRESERVE\_COL\_NAMES=YES option in your LIBNAME statement. Here is an example.

```
libname mydblib oracle user=myusr1 password=mypwd1
    preserve_col_names=yes;
proc sql dquote=ansi;
    create table mydblib.mytable ("my$column" int);
```

## PRESERVE\_COMMENTS= LIBNAME Statement Option

Specifies whether comments should be kept and passed down to your data source for processing.

Valid in: SAS/ACCESS LIBNAME statement, PROC SQL CONNECT statement

Category: Data Set Control

Default: NO

Data source: DB2, Microsoft SQL Server, ODBC, Oracle

Note: Support for this option was added in SAS 9.4M8.

## Syntax

**PRESERVE\_COMMENTS=YES | NO**

### Required Argument

**YES | NO**

specifies whether comments should be passed down to your data source.

## Details

When PRESERVE\_COMMENTS=YES, comments can be passed in an explicit SQL query to your data source for processing. The comment is passed from the CONNECT statement in PROC SQL. The comment that you specify can contain database-specific keywords (hints), special characters, and markup text that are used by your native query processor.

To see the query that is passed to your data source, including any comments that you add, specify SASTRACE=',,d' in the OPTIONS statement.

You can also pass comments for implicit SQL queries that SAS generates automatically, such as for PROC MEANS or PROC PRINT. This capability is available for the data sources that use the PRESERVE\_COMMENT= LIBNAME option. For more information, see “[Macro Variables for Passing Content to DBMS Queries](#)” on page 657.

## Examples

### Example 1: Pass Content via a PROC SQL Query

Here is a query that you pass to your DB2 data source where you set PRESERVE\_COMMENTS= in the CONNECT statement. The query includes a comment that is passed to the data source for processing.

```
options SASTRACE=',,d' msglevel=i;

proc sql ;
  connect to db2(user=myuser pw=mypwd database=mydb2dbms
PRESERVE_COMMENTS=YES) ;
  select * from connection to db2 (
    select Lname, Fname, City, State, IdNumber, Salary, Jobcode
    from staff left join payroll
    on idnumber=idnum
  /*
```

```

<OPTGUIDELINES>
<REGISTRY>
<OPTION NAME='DB2_UNION_OPTIMIZATION'
        VALUE='DISABLE_OJPPD_FOR_NP' />
</REGISTRY>
</OPTGUIDELINES>
; */
)
;
quit ;

```

## Example 2: Pass Content by Using the LIBNAME Option with a PROC SQL Query

Here is a LIBNAME statement and query that you pass to the DB2 data source. The query includes a comment that is passed to the data source for processing.

```

options SASTRACE=',,d' msglevel=i;

libname mydblib db2 dsn=dsnname user=myuser pwd=mypwdval
PRESERVE_COMMENTS=YES;

proc sql ;
  connect using mydblib;
  select * from connection to mydblib (
    select Lname, Fname, City, State, IdNumber, Salary, Jobcode
    from staff left join payroll
    on idnumber=idnum

/*
<OPTGUIDELINES>
<REGISTRY>
<OPTION NAME='DB2_UNION_OPTIMIZATION'
        VALUE='DISABLE_OJPPD_FOR_NP' />
</REGISTRY>
</OPTGUIDELINES>
; */
)
;
quit ;

```

## PRESERVE\_GUID= LIBNAME Statement Option

Preserves the Microsoft SQL Server GUID (Unique Identifier).

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Access

Defaults: YES [ODBC, Microsoft SQL Server]

NO [OLE DB]

NO

Data source: ODBC, OLE DB, Microsoft SQL Server

Note: Support for Microsoft SQL Server and ODBC was added in SAS 9.4M7.

---

## Syntax

PRESERVE\_GUID=[YES](#) | [NO](#)

### Syntax Description

#### **YES**

removes the brackets from around the GUID.

#### **NO**

retains the brackets that are around the GUID.

---

## Details

SAS/ACCESS can deliver Microsoft SQL Server GUIDs (unique identifiers) into SAS with or without curly brackets.

- ODBC and Microsoft SQL Server: 6F9619FF-8B86-D011-B42D-00C04FC964FF
- OLE DB: {6F9619FF-8B86-D011-B42D-00C04FC964FF}

For Microsoft SQL Server and ODBC interfaces, the default is to remove the brackets from the GUID when it is read into SAS.

For OLE DB, the brackets must be removed from the GUIDs values before you execute a join. If PRESERVE\_GUID=NO is specified, the join fails. See the log below for [PRESERVE\\_GUID=NO on page 286](#). Use PRESERVE\_GUID=YES to remove the brackets. See the log for [PRESERVE\\_GUID=YES on page 288](#).

**Example Code 12.1 PRESERVE\_GUID=NO Log**

```

166  proc sql;
167      connect to oledb (
168          init_string=
169              Provider=SQLOLEDB.1;
170              Password=secret;
171              Persist Security Info=True;
172              User ID=user_id;
173              Initial Catalog=Pubs;
174              data source=your_data_source"
175      );
176
177      execute (drop table t) by oledb;
178      execute (CREATE TABLE T(I INT, U uniqueidentifier)) by oledb;
179      execute (INSERT T VALUES(1, '6F9619FF-8B86-D011-B42D-00C04FC964FF')) by
180      oledb;
181
182      disconnect from oledb;
183      quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.04 seconds
      cpu time          0.01 seconds

184
185  libname mylib oledb
186      init_string=
187          Provider=SQLOLEDB.1;
188          Password=secret;
189          Persist Security Info=True;
190          User ID=user_id;
191          Initial Catalog=Pubs;
192          data source=your_data_source" preserve_guid=no ;
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:          OLEDB
      Physical Name:
193
194 /* if preserve_guid=no then the guid value should be in brackets {} */
195 ex: insert into mylib.t values (2, '{7F9619FF-8B86-D011-B42D-
00C04FC964FF}');
196
197 if preserve_guid=yes then the guid value should not be in curly brackets {}
198 ex: insert into mylib.t values (2, '7F9619FF-8B86-D011-B42D-00C04FC964FF');
199 */
200
201
202  proc sql;
203      insert into mylib.t values (2, '{7F9619FF-8B86-D011-B42D-00C04FC964FF'});
NOTE: 1 row was inserted into MYLIB.t.

204  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.01 seconds
      cpu time          0.00 seconds

205
206  proc print data=mylib.t;
207  run;

NOTE: There were 2 observations read from the data set MYLIB.t.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time          0.01 seconds

```

```
208
209  proc contents data=mylib.t;
210  run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time            0.04 seconds
      cpu time             0.00 seconds

211
212  data x;
213    set mylib.t;
214  run;

NOTE: There were 2 observations read from the data set MYLIB.t.
NOTE: The data set WORK.X has 2 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time            0.06 seconds
      cpu time             0.00 seconds

215
216  proc append data=x base=mylib.t;
217  run;

NOTE: Appending WORK.X to MYLIB.t.
NOTE: There were 2 observations read from the data set WORK.X.
NOTE: 2 observations added.
NOTE: The data set MYLIB.t has . observations and 2 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time            0.06 seconds
      cpu time             0.01 seconds

218
219  proc print data=mylib.t;
220  run;

NOTE: There were 4 observations read from the data set MYLIB.t.
NOTE: PROCEDURE PRINT used (Total process time):
      real time            0.00 seconds
      cpu time             0.00 seconds
```

**Example Code 12.2 PRESERVE\_GUID=YES Log**

```

1   proc sql;
2     connect to oledb (
3       init_string=
4         Provider=SQLOLEDB.1;
5         Password=secret;
6         Persist Security Info=True;
7         User ID=user_id;
8         Initial Catalog=Pubs;
9         data source=your_data_source"
10    );
11
12   execute (drop table t) by oledb;
13   execute (CREATE TABLE T(I INT, U uniqueidentifier)) by oledb;
14   execute (INSERT T VALUES(1, '6F9619FF-8B86-D011-B42D-00C04FC964FF')) by
15   oledb;
16
17   disconnect from oledb;
18 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.18 seconds
      cpu time          0.15 seconds

19
20 libname mylib oledb
21   init_string=
22     Provider=SQLOLEDB.1;
23     Password=secret;
24     Persist Security Info=True;
25     User ID=user_id;
26     Initial Catalog=Pubs;
27     data source=your_data_source" preserve_guid=yes ;
NOTE: Libref MYLIB was successfully assigned as follows:
      Engine:          OLEDB
      Physical Name:
28
29 /* if preserve_guid=no then the guid value should be in curly brackets {}
30 ex: insert into mylib.t values (2, '{7F9619FF-8B86-D011-B42D-
00C04FC964FF}');
31
32 if preserve_guid=yes then the guid value should not be in curly brackets {}
33 ex: insert into mylib.t values (2, '7F9619FF-8B86-D011-B42D-00C04FC964FF');
34 */
35
36
37 proc sql;
38   insert into mylib.t values (2, '7F9619FF-8B86-D011-B42D-00C04FC964FF');
NOTE: 1 row was inserted into MYLIB.t.

39 quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time          0.06 seconds
      cpu time          0.01 seconds

40
41 proc print data=mylib.t;
42 run;

NOTE: There were 2 observations read from the data set MYLIB.t.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.03 seconds
      cpu time          0.03 seconds

```

```

43   proc contents data=mylib.t;
44   run;

NOTE: PROCEDURE CONTENTS used (Total process time):
      real time          0.06 seconds
      cpu time           0.03 seconds

46
47   data x;
48   set mylib.t;
49   run;

NOTE: There were 2 observations read from the data set MYLIB.t.
NOTE: The data set WORK.X has 2 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time          0.07 seconds
      cpu time           0.04 seconds

50
51   proc append data=x base=mylib.t;
52   run;

NOTE: Appending WORK.X to MYLIB.t.
NOTE: There were 2 observations read from the data set WORK.X.
NOTE: 2 observations added.
NOTE: The data set MYLIB.t has . observations and 2 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time          0.06 seconds
      cpu time           0.03 seconds

53
54   proc print data=mylib.t;
55   run;

NOTE: There were 4 observations read from the data set MYLIB.t.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.01 seconds
      cpu time           0.00 seconds

```

---

## PRESERVE\_TAB\_NAMES= LIBNAME Statement Option

Preserves spaces, special characters, and case sensitivity in DBMS table names.

|           |                                                                                                                                                                                                                                                                         |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                            |
| Category: | Data Set Control                                                                                                                                                                                                                                                        |
| Alias:    | PRESERVE_NAMES= [see “Details”]                                                                                                                                                                                                                                         |
| Defaults: | YES [Amazon Redshift, Aster, DB2 under z/OS, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, OLE DB, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick]<br>NO [DB2 under UNIX and PC Hosts, Hadoop, JDBC, Netezza, Oracle, PostgreSQL, Spark] |

|              |                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | Varies [ODBC, see “Details”]                                                                                                                                                                                                                                                                                                                                                                 |
| Interaction: | If you use the DS2 or FedSQL language, quoting and casing of names is different. For more information, see the identifiers topic in <a href="#">SAS DS2 Language Reference</a> or <a href="#">SAS FedSQL Language Reference</a> .                                                                                                                                                            |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PC Files, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                                                   |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Impala and JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Spark was added in SAS 9.4M7.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">PRESERVE_COL_NAMES= LIBNAME option</a> , <a href="#">PRESERVE_TAB_NAMES= data set option</a> , <a href="#">DBINDEX= data set option</a> , <a href="#">SAS/ACCESS naming</a> , <a href="#">SCHEMA= LIBNAME option</a> , naming conventions in the DBMS-specific reference section for your SAS/ACCESS interface                                                                   |

## Syntax

**PRESERVE\_TAB NAMES=YES | NO**

## Syntax Description

### YES

specifies that table names are read from and passed to the DBMS with special characters, and the exact, case-sensitive spelling of the name is preserved.

**SAP HANA:** To use reserved words when naming a table for output to the database, you must specify PRESERVE\_TAB NAMES=YES.

### NO

specifies that when you create DBMS tables or refer to an existing table, the table names are derived from SAS member names by using SAS member name normalization. However, the database applies DBMS-specific normalization rules to the SAS member names. Therefore, the table names are created or referenced in the database following the DBMS-specific normalization rules.

When you use SAS to read a list of table names, tables with names that do not conform to SAS member name normalization rules do not appear in output. In SAS line mode, here is how SAS indicates the number of tables that are not displayed from PROC DATASETS because of this restriction:

```
Due to the PRESERVE_TAB NAMES=NO LIBNAME option value,
12 table(s) have not been displayed.
```

You do not receive this warning when you use SAS Explorer. SAS Explorer displays DBMS table names in capitalized form when PRESERVE\_TAB NAMES=NO. This is now how the tables are represented in the DBMS.

NO is the default for most DBMS interfaces.

## Details

To use table names in your SAS program that are not valid SAS names, use one of these techniques.

- Use the PROC SQL option DQUOTE= and place double quotation marks around the table name. The libref must specify PRESERVE\_TAB\_NAMES=YES. Here is an example.

```
libname mydblib oracle user=myusr1 password=mypwd1
      preserve_tab_names=yes;
proc sql dquote=ansi;
  select * from mydblib."my table";
```

- Use name literals in the SAS language. The libref must specify PRESERVE\_TAB\_NAMES=YES. Here is an example.

```
libname mydblib oracle user=myusr1
      password=mypwd1 preserve_tab_names=yes;
proc print data=mydblib.'my table'n;
run;
```

To save some time when coding, specify the PRESERVE\_NAMES= alias if you plan to specify both the PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options in your LIBNAME statement.

*Greenplum, HAWQ:* Unless you specify PRESERVE\_TAB\_NAMES=YES, the table name that you enter is converted to lowercase.

*Hadoop, Spark:* Hive does not preserve case sensitivity and forces all identifiers to be lowercase. However, when PRESERVE\_TAB\_NAMES=NO (the default value), a Hive table name is modified by SAS to be uppercase to match SAS naming rules. If you want to preserve the case of the table name as it is in Hive, then specify PRESERVE\_TAB\_NAMES=YES. SAS then preserves lowercase table names for Hive tables.

*ODBC:* The default for ODBC depends on the case sensitivity of the underlying ODBC driver.

*Oracle:* Unless you specify PRESERVE\_TAB\_NAMES=YES, the table name that you enter for the SCHEMA= LIBNAME option or for the DBINDEX= data set option is converted to uppercase.

## Example

If you use PROC DATASETS to read the table names in an Oracle database that contains three tables, My\_Table, MY\_TABLE, and MY TABLE. The results differ depending on the value of PRESERVE\_TAB\_NAMES.

If the libref specifies PRESERVE\_TAB\_NAMES=NO, the PROC DATASETS output is one table name, MY\_TABLE. This is the only table name that is in Oracle normalized form (uppercase letters and a valid symbol, the underscore). My\_Table is not displayed because it is not in a form that is normalized for Oracle. MY TABLE

is not displayed because it is not in SAS member normalized form: The embedded space is a nonstandard SAS character.

If the libref specifies PRESERVE\_TAB\_NAMES=YES, the PROC DATASETS output includes all three table names: My\_Table, MY\_TABLE, and MY TABLE.

## PRESERVE\_USER= LIBNAME Statement Option

Preserves the case of the value in the USER= connection option.

|               |                                                                                                                                                                                               |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                  |
| Category:     | Data Access                                                                                                                                                                                   |
| Default:      | NO                                                                                                                                                                                            |
| Interactions: | Use the SAS_DB2_PRESERVE_USER, SAS_NETEZZA_PRESERVE_USER, or SAS_ODBC_PRESERVE_USER environment variable for the SAS session.<br>The LIBNAME option value overrides the environment variable. |
| Data source:  | DB2 under UNIX and PC Hosts, ODBC, Netezza                                                                                                                                                    |
| Notes:        | Support for this LIBNAME option was added in SAS 9.4M1.<br>Support for Netezza was added in SAS 9.4M4.                                                                                        |
| Examples:     | Set the LIBNAME option:                                                                                                                                                                       |

```
libname mydblib db2 user=myuser1 pass=mypwd1 PRESERVE_USER=YES;
```

Specify the appropriate environment variable on PC hosts in the Advanced system settings:

```
SAS_DB2_PRESERVE_USER YES  
SAS_NETEZZA_PRESERVE_USER YES  
SAS_ODBC_PRESERVE_USER YES
```

Export the appropriate environment variable on UNIX hosts for the Bourne shell:

```
export SAS_DB2_PRESERVE_USER=YES  
export SAS_NETEZZA_PRESERVE_USER=YES  
export SAS_ODBC_PRESERVE_USER=YES
```

Export the appropriate environment variable on UNIX hosts for the C shell:

```
setenv SAS_DB2_PRESERVE_USER=YES  
setenv SAS_NETEZZA_PRESERVE_USER=YES  
setenv SAS_ODBC_PRESERVE_USER=YES
```

Specify the appropriate environment variable at SAS invocation for UNIX and PC hosts:

```
sas -set SAS_DB2_PRESERVE_USER YES  
sas -set SAS_NETEZZA_PRESERVE_USER YES  
sas -set SAS_ODBC_PRESERVE_USER YES
```

---

## Syntax

**PRESERVE\_USER=YES | NO**

### Required Arguments

#### YES

specifies that SAS/ACCESS preserves the case of the value for the USER= option.

#### NO

specifies that SAS/ACCESS changes the value for the USER= option to uppercase for use in later connections.

---

## Details

DB2 and Netezza typically accept user names that are not case sensitive. Similarly, the ODBC engine can be used to connect to databases that accept user names that are not case sensitive. However, some authentication protocols, such as LDAP, allow case-sensitive user names. When you use such protocols with your database, you can specify PRESERVE\_USER=YES so that the SAS/ACCESS engine retains the original case of the USER= option.

---

## PROGRAM\_NAME= LIBNAME Statement Option

Specifies the string to use as the application identifier for DB2 monitoring.

|              |                                                                                             |
|--------------|---------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                |
| Category:    | Data Set Control                                                                            |
| Alias:       | CORRELATION_ID                                                                              |
| Default:     | none                                                                                        |
| Data source: | DB2 under UNIX and PC Hosts                                                                 |
| Note:        | Support for this LIBNAME option was added for SAS 9.4.                                      |
| Example:     | <code>LIBNAME db2data DB2 DB=sample USER=db2 PWD=db2pwd PROGRAM_NAME='SAS on wrks1';</code> |

---

## Syntax

**PROGRAM\_NAME='user-defined-string'**

## Details

The string that you specify overrides the default application identifier that DB2 chooses. If you are connecting to a host DB2 system through DB2 Connect, the first 12 characters of this string are used as the correlation ID on the host.

---

## PROPERTIES= LIBNAME Statement Option

Specifies JDBC custom connection properties, which override the default JDBC connection properties.

|               |                                                                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME and CONNECT statements                                                                                                                                        |
| Category:     | JDBC connection options                                                                                                                                                          |
| Default:      | none                                                                                                                                                                             |
| Restrictions: | If you are using Cloudera Hadoop, you can specify only one JDBC connection property when HiveServer2 is prior to Hive 0.11.<br>Spark 2.2 does not support this option.           |
| Interaction:  | This option is ignored when you also specify the URI= connection option. For more information, see <a href="#">Arguments for Hadoop</a> or <a href="#">Arguments for Spark</a> . |
| Data source:  | Hadoop, Spark                                                                                                                                                                    |
| Notes:        | Support for this LIBNAME option was added in SAS 9.4M2.<br>Support for Spark was added in SAS 9.4M7.                                                                             |

---

## Syntax

**PROPERTIES='***JDBC-connection-property-1< ;JDBC-connection-property-2... >*

### Syntax Description

#### ***JDBC-connection-property***

specifies one or more JDBC connection options to override the default JDBC connection options.

---

## Details

When you specify JDBC connection properties using the PROPERTIES= LIBNAME option, the properties are appended to the JDBC URL. This overrides the default properties. Site-wide Hive properties are specified in the `hive-site.xml` file in the Hive configuration directory. In the JDBC URL, custom properties are separated from the default properties by the question mark ( ? ) character. The ? denotes the start of Hive configuration options. You do not need to add the ? character in the PROPERTIES= LIBNAME option. To specify Hive variables, add the character # before the Hive variable in the LIBNAME option.

## Examples

### Example 1: Specify Strict Mode

```
libname h4 hadoop schema=sample user=hdusr1 server="hdp2ga"
      properties='hive.mapred.mode=strict';
```

Here is the resulting URL.

```
jdbc:hive2://hdp2ga.unx.sas.com:10000?hive.mapred.mode=strict
```

### Example 2: Specify Strict Mode with a Second Option

```
libname h4 hadoop schema=sample user=hdusr1 server="hdp2ga"
      properties='hive.mapred.mode=strict;hive.optimize.groupby=false';
```

Here is the resulting URL.

```
jdbc:hive2://hdp2ga.unx.sas.com:10000?hive.mapred.mode=strict;
      hive.optimize.groupby=false
```

### Example 3: Specify a Hive Variable

```
libname h4 hadoop schema=sample user=hdusr1 server="hdp2ga"
      properties='#D_TBL=dummy_t';
```

Here is the resulting URL.

```
jdbc:hive2://hdp2ga.unx.sas.com:10000?#D_TBL=dummy_t
```

### Example 4: Specify Strict Mode and a Hive Variable

```
libname h4 hadoop schema=sample user=hdusr1 server="hdp2ga"
      properties='hive.mapred.mode=strict#D_TBL=dummy_t';
```

Here is the resulting URL.

```
jdbc:hive2://hdp2ga.unx.sas.com:10000?hive.mapred.mode=strict#D_TBL=dummy_t
```

### Example 5: Specify Strict Mode and a Hive Principal

```
libname h4 hadoop principal=hive/HiveServer2Host@YOUR-REALM.COM
      schema=sample user=hdusr1 server="hdp2ga"
      properties='hive.mapred.mode=strict#D_TBL=dummy_t';
```

Here is the resulting URL.

```
jdbc:hive2://hdp2ga.unx.sas.com:10000;
      principal=
      hive/HiveServer2Host@YOUR-REALM.COM?hive.mapred.mode=strict#D_TBL=dummy_t
```

---

## PROXY= LIBNAME Statement Option

Specifies the URL of a proxy server that is used to connect to Google BigQuery.

|              |                                                                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                           |
| Category:    | Data Access                                                                                                                                                                                                                                                                            |
| Default:     | none                                                                                                                                                                                                                                                                                   |
| Restriction: | After you have made a connection to Google BigQuery, you cannot alter the PROXY= value unless you start a new SAS session. This also means you cannot specify a new LIBNAME statement that includes the PROXY= option if you have already connected to Google BigQuery without PROXY=. |
| Data source: | Google BigQuery                                                                                                                                                                                                                                                                        |
| Note:        | Support for this option was added in the April 2020 update for SAS/ACCESS on SAS 9.4M6 and SAS Viya 3.5.                                                                                                                                                                               |
| Example:     | <pre>proxy="my_company.proxy.htm" proxy="http://10.99.44.64:4321"</pre>                                                                                                                                                                                                                |

---

## Syntax

**PROXY="*URL*"**

### Syntax Description

***URL***

specifies the URL of the proxy server that is used to connect to Google BigQuery.

---

## QUALIFIER= LIBNAME Statement Option

Allows identification of tables and views with the specified qualifier.

|              |                                                                                                                                                                     |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                        |
| Category:    | Data Access                                                                                                                                                         |
| Alias:       | CATEGORY= [Google BigQuery]                                                                                                                                         |
| Default:     | none                                                                                                                                                                |
| Data source: | Amazon Redshift, Google BigQuery, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, PostgreSQL, Vertica                                                           |
| Notes:       | <p>Support for Amazon Redshift and PostgreSQL was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery was added in the August 2019 release of SAS/ACCESS.</p> |

See: [QUALIFIER= data set option](#)

Example: In a LIBNAME statement:

```
qualifier='cat1'
```

## Syntax

**QUALIFIER=<'><qualifier-name><'>**

## Details

If you omit this option, the default is the default DBMS qualifier name, if any. You can use QUALIFIER= for any DBMS that allows three-part identifier names, such as *qualifier.schema.object*.

*Google BigQuery*: Specify the qualifier in single or double quotation marks. QUALIFIER= is typically the catalog name, not the native project ID. If you are accessing files in the default login project, do not specify a catalog or specify the qualifier to match the libref value. (See the Google BigQuery example below.)

*Microsoft SQL Server*: The Microsoft SQL Server interface supports three-part table names, in the structure *qualifier.schema.table*. It is possible to specify a value for QUALIFIER= and not for SCHEMA=. In this case, a table would be specified as *qualifier..table* in the SQL code that is passed to the database.

*MySQL*: The MySQL interface does not support three-part identifier names, so a two-part name is used (such as *qualifier.object*).

## Examples

### Example 1

In this LIBNAME statement, the QUALIFIER= option causes ODBC to interpret any reference to Mydblib.Employee in SAS as Mydept.Scott.Employee.

```
libname mydblib odbc dsn=myoracle
      password=testpass schema=scott
      qualifier=mydept;
```

### Example 2

In this example, the QUALIFIER= option causes OLE DB to interpret any reference in SAS to Mydblib.Employee as Pcdvision.Raoul.Employee.

```
libname mydblib oledb provider=SQLOLEDB
```

```

properties=("user id"=dbajorge "data source"=SQLSERVR)
schema=raoul qualifier=pcdivision;
proc print data=mydblib.employee;
run;

```

---

## Example 3

For Google BigQuery, if you are accessing files in the default login project, do not specify a catalog or specify the qualifier to match the libref value. In this example, the QUALIFIER= value matches the libref value 'gbq':

```

libname gbq bigquery project='sas-plt'
qualifier='gbq'
schema='sample';

```

---

## QUALIFY\_ROWS= LIBNAME Statement Option

Uniquely qualifies all member values in a result set.

|              |                                                |
|--------------|------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                   |
| Category:    | Data Set Control                               |
| Default:     | NO                                             |
| Data source: | OLE DB                                         |
| See:         | <a href="#">Accessing OLE DB for OLAP Data</a> |

---

## Syntax

**QUALIFY\_ROWS=**[YES](#) | NO

### Syntax Description

#### YES

specifies that when the OLE DB interface flattens the result set of an MDX command, the values in each column are uniquely identified using a hierarchical naming scheme.

#### NO

specifies that when the OLE DB interface flattens the result set of an MDX command, the values in each column are not qualified, which means they might not be unique.

## Details

For example, when this option is set to NO, a GEOGRAPHY column might have a value of PORTLAND for Portland, Oregon, and the same value of PORTLAND for Portland, Maine. When you set this option to YES, the two values might become [USA]. [Oregon]. [Portland] and [USA]. [Maine]. [Portland], respectively.

---

**Note:** Depending on the size of the result set, QUALIFY\_ROWS=YES can have a significant, negative impact on performance. This can occur because it forces the OLE DB interface to search through various schemas to gather the information needed to create unique qualified names.

---

## QUERY\_BAND= LIBNAME Statement Option

Specifies whether to set a query band for the current session.

|              |                                                                                                                                                                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                        |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                    |
| Default:     | none                                                                                                                                                                                                                                                                                |
| Requirement: | A semi-colon ( ; ) is required before the ending quotation mark in order for the database to process "pair-name=pair_value;"                                                                                                                                                        |
| Data source: | Teradata                                                                                                                                                                                                                                                                            |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">FASTEXPORT= LIBNAME option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">QUERY_BAND= data set option</a> |

## Syntax

**QUERY\_BAND="pair-name=pair\_value;"**

### Syntax Description

***pair-name=pair\_value***

specifies a name and value pair of a query band for the current session.

## Details

Use this option to set unique identifiers in Teradata sessions and to add them to the current session. The Teradata engine uses this syntax to pass the name-value pair to Teradata:

```
libname db teradata user=myusr1 password=mypwd1
```

```
QUERY_BAND="org=Marketing;report=Mkt4Q08;" ;
```

For more information about this option and query-band limitations, see *Teradata SQL Reference: Data Definition Statements*.

## QUERY\_TIMEOUT= LIBNAME Statement Option

Specifies the number of seconds of inactivity to wait before canceling a query.

|              |                                                                                                                                                                                                                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME, CONNECT statement                                                                                                                                                                                                                                                                                                       |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                            |
| Alias:       | TIMEOUT= [Greenplum, HAWQ, Microsoft SQL Server, SAP IQ]                                                                                                                                                                                                                                                                                    |
| Default:     | 0                                                                                                                                                                                                                                                                                                                                           |
| Interaction: | Set the SAS SQLIPONEATTEMPT system option to disable PROC SQL from trying the query again after the first query times out.                                                                                                                                                                                                                  |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, Netezza, ODBC, PostgreSQL, SAP IQ, Snowflake, Vertica, Yellowbrick                                                                                                                                                        |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Hadoop and JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">QUERY_TIMEOUT= data set option</a> , <a href="#">SQLIPONEATTEMPT system option</a>                                                                                                                                                                                                                                              |

## Syntax

**QUERY\_TIMEOUT=***number-of-seconds*

### Syntax Description

#### ***number-of-seconds***

specifies a positive integer for the number of seconds to wait before canceling the query. The default value of 0 indicates that there is no time limit for a query. This option is useful when you are testing a query or if you suspect that a query might contain an endless loop.

## QUOTE\_CHAR= LIBNAME Statement Option

Specifies which quotation mark character to use when delimiting identifiers.

|           |                              |
|-----------|------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement |
| Category: | Data Set Control             |

|              |                                                                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Default:     | none                                                                                                                                                                                                                                             |
| Data source: | Amazon Redshift, Aster, Greenplum, HAWQ, Impala, JDBC, Microsoft SQL Server, Netezza, ODBC, OLE DB, PostgreSQL, SAP HANA, SAP IQ, Vertica, Yellowbrick                                                                                           |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added for SAS Viya 3.5.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |

## Syntax

**QUOTE\_CHAR="***character***"**

### Syntax Description

#### *character*

specifies the quotation mark character to use when delimiting identifiers, such as the double quotation mark ("").

Enclose the character in single or double quotation marks, as appropriate. For more information, see the examples below.

## Details

The provider usually specifies the delimiting character. However, when there is a difference between what the provider and the DBMS allow for this character, the QUOTE\_CHAR= option overrides the character that the provider returns.

*Microsoft SQL Server:* QUOTE\_CHAR= overrides the Microsoft SQL Server default.

*ODBC:* This option is mainly for the ODBC interface to SAP ASE, and you should use it with the **DBCONINIT** and **DBLIBINIT** LIBNAME options. QUOTE\_CHAR= overrides the ODBC default because some drivers return a blank for the identifier delimiter even though the DBMS uses a quotation mark (for example, ODBC to SAP ASE).

## Examples

### Example 1: Specify a Single Quotation Mark

Here is what to specify if you want your quotation character to be a single quotation mark.

```
libname x odbc dsn=mydsn pwd=mypassword quote_char='';
```

## Example 2: Specify a Double Quotation Mark

Here is what to specify if you want your quotation character to be a double quotation mark.

```
libname x odbc dsn=mydsn pwd=mypassword quote_char='";
```

## QUOTED\_IDENTIFIER= LIBNAME Statement Option

Allows specification of table and column names with embedded spaces and special characters.

|              |                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                            |
| Category:    | Data Set Control                                                                                        |
| Default:     | NO                                                                                                      |
| Data source: | SAP ASE                                                                                                 |
| See:         | <a href="#">PRESERVE_COL_NAMES= LIBNAME option</a> , <a href="#">PRESERVE_TAB_NAMES= LIBNAME option</a> |

## Syntax

**QUOTED\_IDENTIFIER=YES | NO**

## Details

You use this option in place of the [PRESERVE\\_COL\\_NAMES=](#) and [PRESERVE\\_TAB\\_NAMES= LIBNAME options](#). They have no effect on the SAP ASE interface because it defaults to case sensitivity.

## READ\_ISOLATION\_LEVEL= LIBNAME Statement Option

Specifies the degree of isolation of the current application process from other concurrently running application processes.

|           |                                                               |
|-----------|---------------------------------------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement                                  |
| Category: | Data Set Control                                              |
| Alias:    | RIL= [Greenplum, HAWQ, Microsoft SQL Server, SAP IQ]          |
| Defaults: | COMMITTED READ [Informix]<br>CS [DB2 under UNIX and PC Hosts] |

|                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| determined by the DBMS [DB2 under z/OS]                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| RC [Amazon Redshift, Greenplum, HAWQ, Microsoft SQL Server, ODBC, PostgreSQL, SAP IQ, Vertica, Yellowbrick] |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| READCOMMITTED [Oracle]                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 1 [SAP ASE]                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| none [OLE DB]                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Set the default in the DB2CLI.ini file [ODBC]. For valid values, see your DBMS documentation.               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Interaction:                                                                                                | For DB2 under UNIX and PC Hosts and ODBC, this option is ignored if you do not set the <a href="#">READ_LOCK_TYPE= LIBNAME</a> option to ROW.                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Data source:                                                                                                | Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Informix, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP IQ, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                                                                                                                                                                                           |
| Notes:                                                                                                      | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                                                                                                                                                                                                                                              |
| See:                                                                                                        | <a href="#">CONNECTION= LIBNAME</a> option, <a href="#">READ_ISOLATION_LEVEL= data set option</a> , <a href="#">READ_LOCK_TYPE= LIBNAME</a> option, <a href="#">READ_LOCK_TYPE= data set option</a> , <a href="#">READ_MODE_WAIT= LIBNAME</a> option, <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME</a> option, <a href="#">UPDATE_ISOLATION_LEVEL= data set option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME</a> option, <a href="#">UPDATE_LOCK_TYPE= data set option</a> , and DBMS-specific locking information in the reference section for your SAS/ACCESS interface |

---

## Syntax

**READ\_ISOLATION\_LEVEL=***DBMS-specific value*

---

## Details

The degree of isolation specifies the degree to which these items are affected:

- Rows that the current application reads and updates are available to other concurrently executing applications.
- Update activity of other concurrently executing application processes can affect the current application.

For more information about the values that are accepted, see your DBMS documentation.

---

## READ\_LOCK\_TYPE= LIBNAME Statement Option

Specifies how data in a DBMS table is locked during a READ transaction.

|           |                              |
|-----------|------------------------------|
| Valid in: | SAS/ACCESS LIBNAME statement |
| Category: | Data Access                  |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alias:       | READLOCK_TYPE=                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Defaults:    | none [DB2 under z/OS, Teradata]<br>set by the data provider [OLE DB]<br>NOLOCK [Oracle, SAP ASE]<br>ROW [Amazon Redshift, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Microsoft SQL Server, ODBC, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Vertica, Yellowbrick]                                                                                                                                                                                                                                                                                                                                                                        |
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Notes:       | Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Tip:         | If you omit READ_LOCK_TYPE=, the default is the default action for the DBMS. You can specify a lock for one DBMS table by using the data set option or for a group of DBMS tables by using the LIBNAME option.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| See:         | <a href="#">CONNECTION= LIBNAME option</a> , <a href="#">READ_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">READ_ISOLATION_LEVEL= data set option</a> , <a href="#">READ_LOCK_TYPE= data set option</a> , <a href="#">READ_MODE_WAIT= LIBNAME option</a> , <a href="#">READ_MODE_WAIT= data set option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= data set option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a> , <a href="#">UPDATE_LOCK_TYPE= data set option</a> , and DBMS-specific locking information in the reference section for your SAS/ACCESS interface |

---

## Syntax

**READ\_LOCK\_TYPE=ROW | PAGE | TABLE | NOLOCK | VIEW**

### Syntax Description

#### ROW

locks a row if any of its columns are accessed. If you are using the interface to ODBC or DB2 under UNIX and PC Hosts, READ\_LOCK\_TYPE=ROW indicates that locking is based on the [READ\\_ISOLATION\\_LEVEL= LIBNAME option](#).

**Data source**      Amazon Redshift, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Microsoft SQL Server, ODBC, Oracle, PostgreSQL, SAP HANA, SAP IQ, Teradata, Vertica

#### PAGE

locks a page of data, which is a DBMS-specific number of bytes.

**Data source**    SAP ASE

#### TABLE

locks the entire DBMS table. If you specify READ\_LOCK\_TYPE=TABLE, you must also specify [CONNECTION=UNIQUE](#), or you receive an error message. Specifying CONNECTION=UNIQUE ensures that your table lock is not lost (for example, due to another table closing and committing rows in the same connection).

**Data source** DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, Oracle, SAP IQ, Teradata

**NOLOCK**

does not lock the DBMS table, pages, or rows during a read transaction.

**Data source** Microsoft SQL Server, ODBC with Microsoft SQL Server driver, OLE DB, Oracle, SAP ASE

**VIEW**

locks the entire DBMS view.

**Data source** Teradata

## Example

In this example, the libref MYDBLIB uses SAS/ACCESS Interface to Oracle to connect to an Oracle database. USER=, PASSWORD=, and PATH= are SAS/ACCESS connection options. The LIBNAME options specify to use row-level locking when data is read or updated.

```
libname mydblib oracle user=myusr1 password=mypwd1
path=mysrv1 read_lock_type=row update_lock_type=row;
```

## READ\_METHOD= LIBNAME Statement Option

Specifies how to read data.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: none

Data source: Hadoop, Spark

Note: Support for Spark was added in SAS 9.4M7.

See: [ANALYZE= LIBNAME option](#), [ANALYZE= data set option](#), [READ\\_METHOD= data set option](#)

## Syntax

**READ\_METHOD=**[JDBC | HDFS](#)

## Syntax Description

### JDBC

specifies that data is to be read through the JDBC connection to the Hive service. You can use the ANALYZE= option to potentially improve performance when querying small tables.

### HDFS

specifies that data is to be read through a connection to the Hadoop HDFS service.

## Details

Although HDFS cannot alter the behavior of operations that always use JDBC, in general HDFS is a faster alternative to JDBC. To take advantage of potential performance benefits, set this option to HDFS. Use JDBC when you cannot access the HDFS service or JDBC Read offers some other advantage.

## Example: Read Data Using JDBC

In this example, a partition of data from the sales Hive table is read using JDBC.

```
libname hdp hadoop server=mysrv1 user=myusr1 pwd=mypwd1;
data work.sales_subset; set hdp.sales(READ_METHOD=JDBC);
where year_month='2012-10'; run;
```

## READ\_MODE= LIBNAME Statement Option

Specifies the method to use to move data into SAS.

|              |                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                  |
| Category:    | Data Set Control                                                                                              |
| Default:     | STANDARD                                                                                                      |
| Data source: | Google BigQuery                                                                                               |
| Note:        | Support for this option was added in the April 2021 update for SAS 9.4M7.                                     |
| Tip:         | The STORAGE value is recommended if the Google Storage API is available for your Google BigQuery data source. |
| See:         | <a href="#">READ_MODE= data set option</a>                                                                    |

## Syntax

**READ\_MODE=STANDARD | STORAGE**

## Required Argument

### **STANDARD | STORAGE**

specifies the method to use when moving data from Google BigQuery into SAS.

STANDARD    use the default SAS/ACCESS method to move data into SAS.

STORAGE    use the Storage API for Google to move data into SAS.

## READ\_MODE\_WAIT= LIBNAME Statement Option

During SAS/ACCESS Read operations, specifies whether Teradata should wait to acquire a lock or fail the request when a different user has already locked the DBMS resource.

Valid in:            SAS/ACCESS LIBNAME statement

Category:           Data Access

Default:            none

Data source:        Teradata

See:                [READ\\_ISOLATION\\_LEVEL= LIBNAME option](#), [READ\\_LOCK\\_TYPE= LIBNAME option](#), [READ\\_MODE\\_WAIT= data set option](#), [Locking in the Teradata Interface](#)

## Syntax

**READ\_MODE\_WAIT=YES | NO**

## Syntax Description

### **YES**

specifies for Teradata to wait to acquire the lock, so SAS/ACCESS waits indefinitely until it can acquire the lock.

### **NO**

specifies Teradata fails the lock request if the specified DBMS resource is locked.

## Details

If you specify READ\_MODE\_WAIT=NO and if a different user holds a restrictive lock, the executing SAS step fails. SAS/ACCESS continues processing the job by executing the next step.

If you specify READ\_MODE\_WAIT=YES, SAS/ACCESS waits indefinitely until it can acquire the lock.

A *restrictive* lock means that another user is holding a lock that prevents you from obtaining the lock that you want. Until the other user releases the restrictive lock,

you cannot obtain your lock. For example, another user's table-level WRITE lock prevents you from obtaining a READ lock on the table.

---

## READBUFF= LIBNAME Statement Option

Specifies the number of rows of DBMS data to read into the buffer.

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                                                                                                                                                                                                               |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Aliases:      | BUFF=, BUFFSIZE= [Oracle]<br>ROWSET_SIZE= [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, ODBC, OLE DB, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Vertica]<br>ROWSET= [Aster, Greenplum, HAWQ, SAP IQ]                                                                                                                                                             |
| Default:      | DBMS-specific                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Restrictions: | When READBUFF=1, only one row is retrieved at a time.<br>[Oracle] The default value is calculated based on 250 rows as the row size. 250 rows is the minimum value that this option can accept.                                                                                                                                                                                                                                               |
| Interactions: | Buffering data reads can decrease network activities and increase performance. However, because SAS stores the rows in memory, higher values for READBUFF= use more memory. In addition, if too many rows are selected at once, rows that are returned to the SAS application might be out of date. For example, if someone else modifies the rows, you do not see the changes.<br>Hadoop: This option is applied only when READ_METHOD=JDBC. |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Vertica, Yellowbrick                                                                                                                                                                               |
| Notes:        | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.<br>Support for Hadoop, JDBC, and Spark was added on SAS 9.4M8.                                                                                                 |
| Tip:          | This option improves performance by specifying a number of rows that can be held in memory for input into SAS.                                                                                                                                                                                                                                                                                                                                |
| See:          | <a href="#">INSERTBUFF= LIBNAME option</a> , <a href="#">INSERTBUFF= data set option</a> , <a href="#">READBUFF= data set option</a>                                                                                                                                                                                                                                                                                                          |

---

## Syntax

**READBUFF=***integer*

## Syntax Description

### **integer**

the positive number of rows to hold in memory. SAS allows the maximum number that the DBMS allows.

---

## Details

**Table 12.6 DBMS-Specific Default Values**

| DBMS                        | Default                                                                                                                                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Amazon Redshift             | 0                                                                                                                                                                                                                         |
| PostgreSQL                  |                                                                                                                                                                                                                           |
| Yellowbrick                 |                                                                                                                                                                                                                           |
| Aster                       | automatically calculated based on row length                                                                                                                                                                              |
| Google BigQuery             | <i>Google BigQuery:</i> The maximum value is 32,767.                                                                                                                                                                      |
| Greenplum                   |                                                                                                                                                                                                                           |
| Hadoop                      |                                                                                                                                                                                                                           |
| HAWQ                        |                                                                                                                                                                                                                           |
| Impala                      |                                                                                                                                                                                                                           |
| JDBC                        |                                                                                                                                                                                                                           |
| Netezza                     |                                                                                                                                                                                                                           |
| SAP HANA                    |                                                                                                                                                                                                                           |
| SAP IQ                      |                                                                                                                                                                                                                           |
| Snowflake                   |                                                                                                                                                                                                                           |
| Spark                       |                                                                                                                                                                                                                           |
| DB2 under UNIX and PC Hosts | If you do not specify a value for this option, the default buffer size is automatically calculated based on the row length of your data. The SQLExtendedFetch API call is used.                                           |
| DB2 under z/OS              | For SAS 9.2 and above, the default is 1 and the maximum value is 32,767.<br>For more information, see “ <a href="#">READBUFF= Restriction</a> ” <a href="#">on page 798</a> .                                             |
| Microsoft SQL Server ODBC   | The default is 0. If you do not specify a value for this option, the SQLFetch API call is used and no internal SAS buffering is performed. When you specify READBUFF=1 or greater, the SQLExtendedFetch API call is used. |
| OLE DB                      | 1                                                                                                                                                                                                                         |

| DBMS    | Default                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Vertica |                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Oracle  | The default is 250. If you do not specify this option, the READBUFF= value is automatically calculated as the number of rows that can fit into a memory buffer of 100K on most platforms. On the z/OS platform, it is calculated as the number of rows that can fit into a memory buffer of 50K. However, it is adjusted to be within the range of 250–15000. You can always override the default value by explicitly specifying a value for this option. |
| SAP ASE | The default is 100. To use a value greater than 1 row for this option, you must specify CONNECTION=UNIQUE.                                                                                                                                                                                                                                                                                                                                                |

## REFRESH\_TOKEN= LIBNAME Statement Option

Specifies the refresh token value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Access

Default: none

Interaction: This option is used in conjunction with the CLIENT\_ID= and CLIENT\_SECRET= options.

Data source: Google BigQuery

Note: Support for this option was added in the April 2020 update for SAS/ACCESS.

See: [CLIENT\\_ID= LIBNAME option on page 180](#), [CLIENT\\_SECRET= LIBNAME option on page 180](#)

Example: REFRESH\_TOKEN="1//0dJj ... GA0SNwF-L9I ... 13-7c0oDxCR0p9u6k3\_jl\_i-Y31hdZ4FOumSO0"

## Syntax

**REFRESH\_TOKEN=***token-value*

## Syntax Description

### ***token-value***

specifies the refresh token value for the OAuth client that is used to connect to Google BigQuery using the OAuth authentication method.

---

## Details

This value is masked in the SAS log.

This option accepts values that have been encoded using PROC PWENCODE. SAS/ACCESS recognizes encoded values as those that begin with a SAS encoding tag. For more information, see “[PWENCODE Procedure](#)” in *Base SAS Procedures Guide*.

---

## REMOTE\_DBTYPE= LIBNAME Statement Option

Specifies whether the libref points to a database server on z/OS or to one on Linux, UNIX, or Windows.

|              |                                                                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                             |
| Category:    | Data Set Control                                                                                                                                                                         |
| Default:     | ZOS                                                                                                                                                                                      |
| Restriction: | This option is ignored if you do not use it with either the SERVER= CONNECT statement option or the <a href="#">LOCATION= LIBNAME</a> option.                                            |
| Requirement: | Use this option with the SERVER= CONNECT statement option or the LOCATION= LIBNAME option.                                                                                               |
| Data source: | DB2 under z/OS                                                                                                                                                                           |
| See:         | <a href="#">LOCATION= LIBNAME option</a> , <a href="#">SERVER= CONNECT statement option</a> ( <a href="#">SQL pass-through facility Specifics for DB2 under z/OS - Key Information</a> ) |

---

## Syntax

**REMOTE\_DBTYPE=LUW | ZOS**

### Syntax Description

#### **LUW**

specifies that the database server that is accessed through the libref resides on Linux, UNIX, or Windows (LUW).

#### **ZOS**

specifies that the database server that is accessed through the libref resides on z/OS.

---

## Details

Specifying REMOTE\_DBTYPE= in the LIBNAME statement ensures that the SQL that some SAS procedures use to access the DB2 catalog tables is generated properly and is based on the database server type. It also lets such special catalog

calls as DBMS::Indexes function properly when the target database does not reside on a mainframe computer.

If the target data source is a DB2 LUW or another DB2 database on z/OS, the SQL dictionary is loaded when you specify this option.

---

## Example

This example uses REMOTE\_DBTYPE= with the SERVER= option.

```
libname mylib db2 ssid=db2a server=db2_udb remote_dbtype=luw;
proc datasets lib=mylib;
quit;
```

By specifying REMOTE\_DBTYPE=LUW, this SAS code lets the catalog call work properly for this remote connection.

```
proc sql;
connect to db2 (ssid=db2a server=db2_udb remote_dbtype=luw);
select * from connection to db2
select * from connection to db2
(DBMS::PrimaryKeys ("", "JOSMITH", ""));
quit;
```

---

## REREAD\_EXPOSURE= LIBNAME Statement Option

Specifies whether the SAS/ACCESS engine functions like a random access engine for the scope of the LIBNAME statement.

|              |                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                  |
| Category:    | Data Set Control                                                                                                                                                                                                                                                              |
| Alias:       | READ_EXPOSURE_OK= [Oracle]                                                                                                                                                                                                                                                    |
| Default:     | NO                                                                                                                                                                                                                                                                            |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                          |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a>                                                                                                                                                                     |

## Syntax

**REREAD\_EXPOSURE=YES | NO**

### Syntax Description

#### YES

specifies that the SAS/ACCESS engine functions like a random access engine when rereading a row so that you cannot guarantee that the same row is returned. For example, if you read row 5 and someone else deletes it, you read a different row the next time you read row 5. You have the potential for data integrity exposures within the scope of your SAS session.

#### NO

specifies that the SAS/ACCESS engine functions as a serial engine with limited random access capabilities. In this case, your data is protected by the normal data protection that SAS provides.

## Details

#### CAUTION

Using REREAD\_EXPOSURE= could cause data integrity exposures.

*Netezza, ODBC, OLE DB:* If you set this option to YES, it is advisable to set **UPDATE\_ISOLATION\_LEVEL=S** (serializable) to avoid data integrity problems.

*Oracle:* If you set this option to YES, it is advisable to set **UPDATE\_LOCK\_TYPE=TABLE** to avoid data integrity problems.

## RESULTS= LIBNAME Statement Option

Determines where to store query results.

|              |                                                                                                |
|--------------|------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                   |
| Category:    | Data Set Control                                                                               |
| Default:     | MEMORY                                                                                         |
| Data source: | MySQL                                                                                          |
| Example:     | Send results to a temporary disk file:<br><code>libname spooled mysql ... results=disk;</code> |

## Syntax

**RESULTS=MEMORY | SERVER | DISK**

## Syntax Description

### **MEMORY**

stores query results in client memory.

### **SERVER**

stores query results on the server.

### **DISK**

stores query results in a temporary disk file on the client computer.

## Details

Multiple concurrent connections to the server are not supported. Therefore, when RESULTS=SERVER, the entire query must be one that you can push to the server. If not, this message appears in the SAS log:

Commands out of sync; you can't run this command now.

RESULTS=DISK lets you run complex queries with result sets that would typically cause an out-of-memory error. Result-set size is limited only to the free space on the drive that is used for temporary files.

## SAS\_DBMS\_AUTOMETADATA= LIBNAME Statement Option

Specifies whether to transfer data types and character-set metadata from input to output for DATA step processing.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: NO

Interaction: Use the SAS\_DBMS\_AUTOMETADATA environment variable for the entire SAS session.

Data source: Teradata

Note: Support for this LIBNAME option was added for SAS 9.4.

Examples: Specify the LIBNAME option:

```
libname tdlib teradata user=myuser1 pass=mypwd1 SAS_DBMS_AUTOMETADATA=YES;
```

Specify the environment variable on PC hosts:

```
SAS_DBMS_AUTOMETADATA 1
```

Export the environment variable on UNIX hosts [Bourne and Korn shells]:

```
SAS_DBMS_AUTOMETADATA=1
export SAS_DBMS_AUTOMETADATA
```

Export the environment variable on UNIX hosts [C shell]:

```
export SAS_DBMS_AUTOMETADATA=1
```

Specify the environment variable at SAS invocation for PC, UNIX, and z/OS hosts:

```
sas -set SAS_DBMS_AUTOMETADATA 1
```

## Syntax

**SAS\_DBMS\_AUTOMETADATA=YES | NO**

# SCANSTRINGCOLUMNS= LIBNAME Statement Option

Specifies whether to determine the maximum length of VARCHAR columns in a database table or a query.

|              |                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                             |
| Category:    | Data Set Control                                                                                         |
| Alias:       | SCAN_STRING_COLUMNS=,SCAN_STRINGS=, SCANSTRINGS=                                                         |
| Default:     | NO                                                                                                       |
| Data source: | Google BigQuery                                                                                          |
| Note:        | Support for this option was added in the April 2021 update for SAS/ACCESS on SAS 9.4M7 and SAS Viya 3.5. |
| See:         | <a href="#">SCANSTRINGCOLUMNS= data set option</a>                                                       |

## Syntax

**SCANSTRINGCOLUMNS=YES | NO**

## Required Arguments

### YES

specifies to scan all VARCHAR columns in a database table to determine the actual maximum length of the columns before loading data. If the VARCHAR columns have been created with the maximum VARCHAR precision, then using this option can reduce the size of the resulting table and can accelerate the loading process.

### NO

specifies that VARCHAR columns are not scanned before loading data.

## Details

This option can be specified for any table, but the best performance improvement occurs for larger tables. A table is considered to be large if it contains a large number of rows, a large number of VARCHAR columns, or both.

This option applies to any table that you access or to any query that you are running. For example, if you specify SCANSTRINGCOLUMNS=YES and call PROC PRINT for a table, then the scan runs for the table that you are printing. When you run a PROC SQL query, SCANSTRINGCOLUMNS=YES results in scans being done for any table in the query.

---

## SCHEMA= LIBNAME Statement Option

Specifies the schema to use when accessing tables and views in a database.

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                                                        |
| Category:     | Data Access                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Aliases:      | DATABASE= [Hadoop, Impala]<br>OWNER= [Greenplum, HAWQ, Microsoft SQL Server, SAP IQ, Snowflake]                                                                                                                                                                                                                                                                                                                                     |
| Default:      | DBMS-specific                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Requirements: | Google BigQuery: This LIBNAME option is required to access Google BigQuery.<br>Snowflake: This LIBNAME option is required to access Snowflake.                                                                                                                                                                                                                                                                                      |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                                                                          |
| Notes:        | Support for Netezza was added in SAS 9.4M2.<br>Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for JDBC and MySQL was added in SAS Viya 3.4.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Spark was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:          | <a href="#">DBCONINIT= LIBNAME option</a> , <a href="#">PRESERVE_TAB_NAMES= LIBNAME option</a> , <a href="#">SCHEMA= data set option</a>                                                                                                                                                                                                                                                                                            |

---

## Syntax

**SCHEMA=<'>*schema-name*<'>**

## Syntax Description

### **schema-name**

specifies the name that is assigned to a logical classification of objects in a relational database.

If the schema name contains spaces or non-alphanumeric characters, enclose the value in quotation marks.

---

## Details

To use this option, you must have the appropriate privileges to the specified schema.

If you do not specify this option, you connect to any schema in the default database for your DBMS. When you specify SCHEMA=, this option acts as a filter to access only the tables and views that belong to that schema.

*DB2 under z/OS:* If you specify DBCONINIT="SET CURRENT SQLID='user-ID'", then any value that is specified for SCHEMA= is ignored.

**Table 12.7 SCHEMA= Defaults for Each Database**

| DBMS                 | Default                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Amazon Redshift      | none                                                                                                                                                                                                                                                       |
| Google BigQuery      |                                                                                                                                                                                                                                                            |
| Greenplum            |                                                                                                                                                                                                                                                            |
| HAWQ                 |                                                                                                                                                                                                                                                            |
| JDBC                 |                                                                                                                                                                                                                                                            |
| Microsoft SQL Server |                                                                                                                                                                                                                                                            |
| MySQL                |                                                                                                                                                                                                                                                            |
| ODBC                 |                                                                                                                                                                                                                                                            |
| OLE DB               |                                                                                                                                                                                                                                                            |
| PostgreSQL           |                                                                                                                                                                                                                                                            |
| SAP HANA             |                                                                                                                                                                                                                                                            |
| SAP IQ               |                                                                                                                                                                                                                                                            |
| Vertica              |                                                                                                                                                                                                                                                            |
| Yellowbrick          |                                                                                                                                                                                                                                                            |
| Aster                | none<br><br>This uses the database user's default schema.<br>However, the user name is used instead when the user's default schema is the user name. An example is when SQLTables is called to obtain a table listing using PROC DATASETS or SAS Explorer. |
| DB2                  | none                                                                                                                                                                                                                                                       |

| DBMS      | Default                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           | When SCHEMA= is not specified, the SAS/ACCESS engine for DB2 uses the user ID for operations like PROC DATASETS.                                                                                                                                                                                                                                                                                                    |
| Hadoop    | the Hive schema named <code>default</code>                                                                                                                                                                                                                                                                                                                                                                          |
| Impala    | the Impala schema named <code>default</code>                                                                                                                                                                                                                                                                                                                                                                        |
| Informix  | your user name<br><br>Note: The SCHEMA= LIBNAME option is ignored when you are using implicit pass-through and the Informix engine.                                                                                                                                                                                                                                                                                 |
| Netezza   | your default schema<br><br>Note: Netezza supports three-part names in the form <database>.<schema>.<table>, so you can provide both DATABASE= and SCHEMA= in a Netezza LIBNAME statement.                                                                                                                                                                                                                           |
| Oracle    | Specify a schema name to be used when referring to database objects. SAS can access another user's database objects by using a specified schema name. If <code>PRESERVE_TAB_NAMES=NO</code> , SAS converts the SCHEMA= value to uppercase because all values in the Oracle data dictionary are uppercase unless quoted.                                                                                             |
| SAP ASE   | none<br><br>You cannot use the SCHEMA= option when you use <code>UPDATE_LOCK_TYPE=PAGE</code> to update a table.                                                                                                                                                                                                                                                                                                    |
| Snowflake | PUBLIC                                                                                                                                                                                                                                                                                                                                                                                                              |
| Spark     | the Spark schema named <code>default</code>                                                                                                                                                                                                                                                                                                                                                                         |
| Teradata  | none<br><br>You can use this option to point to a different database. Using the SCHEMA= option does not establish a physical connection to the specified schema. This option lets you view or modify another user's DBMS tables or views if you have the required Teradata privileges. For example, to read another user's tables, you must have the Teradata privilege <code>SELECT</code> for that user's tables. |

## Example

In this example, SCHEMA= causes DB2 to interpret any reference in SAS to mydb.employee as scott.employee.

```
libname mydb db2 SCHEMA=SCOTT;
```

To access an Oracle object in another schema, use the SCHEMA= option, as in this example. The schema name is typically a user name or ID.

```
libname mydblib oracle user=myusr1  
password=mypwd1 path='mysrv1' schema=john;
```

In this example, the Oracle SCHEDULE table resides in the AIRPORTS schema and is specified as AIRPORTS.SCHEDULE. To access this table in PROC PRINT and still use the libref (CARGO) in the SAS/ACCESS LIBNAME statement, specify the schema in the SCHEMA= option. Then put in the *libref.table* the DATA statement for the procedure.

```
libname cargo oracle schema=airports user=myusr1 password=mypwd1  
path="mysrv1";  
proc print data=cargo.schedule;  
run;
```

In this Teradata example, the MYUSR1 user prints the Emp table, which is located in the OTHERUSER database.

```
libname mydblib teradata user=myusr1 pw=mypwd1 schema=otheruser;  
proc print data=mydblib.emp;  
run;
```

---

## SCRATCH\_DB= LIBNAME Option LIBNAME Statement Option

Specifies a Hive schema so that SAS can store temporary output.

|              |                                                 |
|--------------|-------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                    |
| Category:    | Data Access                                     |
| Default:     | none (the current schema is used)               |
| Requirement: | SAS must be able to write to the schema.        |
| Data source: | Hadoop, Impala                                  |
| Note:        | Support for this option was added in SAS 9.4M4. |
| See:         | <a href="#">SCRATCH_DB= data set option</a>     |

---

## Syntax

**SCRATCH\_DB=***schema-name*

## Syntax Description

### **schema-name**

specifies the name that is assigned to a logical classification of objects in a relational database.

## Details

This option lets the user specify a target schema that SAS can use for temporary output. It is needed when a user does not have permissions to perform DDL operations such as CREATE TABLE or DROP TABLE in the current schema.

## Example

```
libname hdp hadoop server=hxduped
      user=myusr1 password=mypwd1 scratch_db="tempdb";
```

## SESSIONS= LIBNAME Statement Option

Specifies how many Teradata sessions to be logged on when using FastLoad, FastExport, or MultiLoad.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: none

Data source: Teradata

See: [FASTEXPORT= LIBNAME option](#), [SESSIONS= data set option](#), [Using MultiLoad](#)

## Syntax

**SESSIONS=***number-of-sessions*

## Syntax Description

### **number-of-sessions**

specifies a numeric value that indicates the number of sessions to be logged on.

## Details

When you read data with [FastExport](#) or load data with [FastLoad](#) or [MultiLoad](#), you can request multiple sessions to increase throughput. Using large values might not

necessarily increase throughput due to the overhead associated with session management. Check whether your site has any recommended value for the number of sessions to use. See your Teradata documentation for details about using multiple sessions.

---

## Example

This example uses SESSIONS= in a LIBNAME statement to request five sessions for loading data with FastLoad.

```
libname x teradata user=myusr1 pw=mypwd1 SESSIONS=5;

proc datasets library=x;
  delete test;run;

data x.test (FASTLOAD=YES) ;
  i=5;
run;
```

---

## SHOW\_SYNONYMS= LIBNAME Statement Option

Specifies whether PROC DATASETS shows synonyms, tables, views, or materialized views for the current user and schema if you specified the SCHEMA= option.

|              |                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                           |
| Category:    | Data Set Control                                                                                                                                                       |
| Default:     | NO                                                                                                                                                                     |
| Data source: | Oracle                                                                                                                                                                 |
| See:         | <a href="#">DBLINK= LIBNAME option</a> , <a href="#">DB_OBJECTS= LIBNAME option</a> , <a href="#">SCHEMA= LIBNAME option</a> , <a href="#">SCHEMA= data set option</a> |

---

## Syntax

**SHOW\_SYNONYMS=YES | NO**

### Syntax Description

#### **YES**

specifies that PROC DATASETS shows only synonyms that represent tables, views, or materialized views for the current user.

#### **NO**

specifies that PROC DATASETS shows only tables, views, or materialized views for the current user.

## Details

Rather than submit PROC DATASETS, you can select the libref in SAS Explorer to obtain this same information. By default, no PUBLIC synonyms are displayed unless you specify [SCHEMA=PUBLIC](#).

When you specify only the SCHEMA option, the current schema is always displayed with the appropriate privileges.

Tables, views, materialized views, or synonyms on the remote database always are displayed when you specify the [DBLINK=LIBNAME](#) option. If a synonym represents an object on a remote database that you might not be able to read, you might receive an Oracle error. An example is a synonym representing a sequence.

Synonyms, tables, views, and materialized views in a different schema are also displayed.

---

## SLEEP= LIBNAME Statement Option

Specifies the number of minutes that FastExport, FastLoad, or MultiLoad waits before trying again to log on to Teradata.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Default:     | 6                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Tip:         | The data set option has precedence over the LIBNAME option.                                                                                                                                                                                                                                                                                                                                                                                                  |
| See:         | <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">FASTEXPORT= LIBNAME option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">SLEEP= data set option</a> , <a href="#">TENACITY= LIBNAME option</a> , <a href="#">TENACITY= data set option</a> , <a href="#">Using the TPT API</a> |

---

## Syntax

**SLEEP=***number-of-minutes*

### Syntax Description

***number-of-minutes***

the number of minutes to wait before trying again to log on to Teradata.

---

## Details

Use this option to indicate to [FastExport](#), [FastLoad](#), or [MultiLoad](#) how long to wait before trying to log on to Teradata again when the maximum number of utilities are

already running. (The maximum number of Teradata utilities that can run concurrently varies from 5 to 15, depending on the database server value.) The default value for SLEEP= is 6 minutes. The value that you specify for SLEEP= must be greater than 0.

Use SLEEP= with [TENACITY=](#). TENACITY= specifies the time in hours that FastExport, FastLoad, or MultiLoad must continue to try the logon operation. SLEEP= and TENACITY= function very much like the SLEEP and TENACITY run-time options of the native Teradata FastExport, FastLoad, or MultiLoad utility.

## SPOOL= LIBNAME Statement Option

Specifies whether SAS creates a utility spool file during Read transactions that read data more than once.

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                         |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:     | YES                                                                                                                                                                                                                                                                                                                                                  |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                  |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">CONNECTION= LIBNAME option</a>                                                                                                                                                                                                                                                                                                           |

## Syntax

**SPOOL=YES | NO | DBMS**

### Syntax Description

#### YES

specifies that SAS creates a utility spool file into which it writes the rows that are read the first time. For subsequent passes through the data, the rows are read from the utility spool file rather than being reread from the DBMS table. This guarantees that the rowset is the same for every pass through the data.

#### NO

specifies that the required rows for all passes of the data are read from the DBMS table. No spool file is written. There is no guarantee that the rowset is the same for each pass through the data.

**DBMS**

specifies that the required rows for all passes of the data are read from the DBMS table. However, additional enforcements are made on the DBMS server side to ensure that the rowset is the same for every pass through the data. This value causes SAS/ACCESS Interface to Oracle to satisfy the two-pass requirement by starting a read-only transaction. SPOOL=YES and SPOOL=DBMS have comparable performance results for Oracle. However, SPOOL=DBMS does not use any disk space. When SPOOL is set to DBMS, you must specify CONNECTION=UNIQUE or an error occurs.

Data source   Oracle

## Details

In some cases, SAS processes data in more than one pass through the same set of rows. Spooling is the process of writing rows that have been retrieved during the first pass of a data Read to a spool file. In the second pass, rows can be reread without performing input and output to the DBMS a second time. When data must be read more than once, spooling improves performance. Spooling also guarantees that the data remains the same between passes, as most SAS/ACCESS interfaces do not support member-level locking.

*MySQL*: Do not use SPOOL=NO with the MySQL interface.

*Teradata*: SPOOL=NO requires SAS/ACCESS to issue identical SELECT statements to Teradata twice. In addition, because the Teradata table can be modified between passes, SPOOL=NO can cause data integrity problems. Use SPOOL=NO with discretion.

## SQL\_FUNCTIONS= LIBNAME Statement Option

Customizes the in-memory SQL dictionary function list for this particular LIBNAME statement.

|               |                                                                                                                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                           |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                       |
| Default:      | none                                                                                                                                                                                                                                                                                   |
| Restrictions: | Informix and OLE DB support only SQL_FUNCTIONS=ALL.<br>You must specify a two-part data set name, such as <libref.member>. Otherwise, an error results.<br><libref.member> must be a SAS data set. No check is performed to ensure that it is assigned to the default Base SAS engine. |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick    |
| Notes:        | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for JDBC and Microsoft SQL Server was added in SAS Viya 3.4.                                                                                     |

Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

See: [SQL\\_FUNCTIONS\\_COPY= LIBNAME option](#), “Passing SAS Functions” section for your specific SAS/ACCESS interface

## Syntax

**SQL\_FUNCTIONS=ALL | EXTERNAL\_REPLACE=<libref.member>  
| EXTERNAL\_APPEND=<libref.member>**

### Syntax Description

#### **ALL**

customizes the in-memory SQL dictionary function list for this particular LIBNAME statement by adding the set of all existing functions, even those that might be risky or untested.

#### **EXTERNAL\_REPLACE=<libref.member>**

indicates a user-specified, external SAS data set from which the complete function list in the SQL dictionary is to be built. The assumption is that the user has already issued a LIBNAME statement to the directory where the SAS data set exists.

**Restriction** This value is not valid for Google BigQuery, Informix, or OLE DB.

#### **EXTERNAL\_APPEND=<libref.member>**

indicates a user-specified, external SAS data set from which additional functions are to be added to the existing function list in the SQL dictionary. The assumption is that the user has already issued a LIBNAME statement to the directory where the SAS data set exists.

**Restriction** This value is not valid for Google BigQuery, Informix, or OLE DB.

## Details

Using this option can cause unexpected results, especially if you use it for NULL processing and for handling date, time, and timestamp. For example, when executed without SQL\_FUNCTIONS= enabled, this SAS code returns the SAS date 15308.

```
proc sql;
  select distinct DATE () from x.test;
quit;
```

However, with SQL\_FUNCTIONS=ALL, the same code returns 2001-1-29, which is an ODBC-specific date format. So, be careful when you use this option.

Functions that are passed are different for each DBMS. See the DBMS-specific reference section for your SAS/ACCESS interface for list of functions that it supports.

Here are additional details to keep in mind when you add to or modify the SAS data set.

**Table 12.8** Caveats When Using the SQL\_FUNCTIONS= Option

| Variable            | Required <sup>1</sup> | Optional <sup>2</sup> | Read-Only <sup>2</sup> | Valid Values                                                                              |
|---------------------|-----------------------|-----------------------|------------------------|-------------------------------------------------------------------------------------------|
| SASFUNCNAME         | •                     |                       |                        | Truncated to 32 characters if length is greater than 32                                   |
| SASFUNCNAMELEN      | •                     |                       |                        | Must correctly reflect the length of SASFUNCNAME                                          |
| DBMSFUNCNAME        | •                     |                       |                        | Truncated to 50 characters if length is greater than 50                                   |
| DBMSFUNCNAMELEN     | •                     |                       |                        | Must correctly reflect the length of DBMSFUNCNAME                                         |
| FUNCTION_CATEGORY   |                       | •                     |                        | AGGREGATE , CONSTANT, SCALAR                                                              |
| FUNC_USAGE_CONTEXT  |                       | •                     |                        | SELECT_LIST, WHERE_ORDERBY                                                                |
| FUNCTION_RETURNTYPE |                       | •                     |                        | BINARY, CHAR, DATE, DATETIME, DECIMAL, GRAPHIC, INTEGER, INTERVAL, NUMERIC, TIME, VARCHAR |
| FUNCTION_NUM_ARGS   |                       | •                     |                        | 0                                                                                         |
| CONVERT_ARGS        |                       |                       | •                      | Must be set to 0 for a newly added function.                                              |
| ENGINEINDEX         |                       |                       | •                      | Must remain unchanged for existing functions. Set to 0 for a newly added function.        |

<sup>1</sup> An error results when a value is missing.

<sup>2</sup> For new and existing functions.

## Examples

### Example 1: Include and Replace Existing Functions

You can use EXTERNAL\_APPEND= to include one or more existing functions to the in-memory function list and EXTERNAL\_REPLACE= to replace them. In this example, the DATEPART function in a SAS data set of Oracle functions by appending the function to an existing list of SAS functions.

```
proc sql;
```

```

create table work.append as select *
  from work.allfuncs where sasfuncname='DATEPART';
quit;

libname mydblib oracle sql_functions="EXTERNAL_APPEND=work.append"
  sql_functions_copy=saslog;

```

## Example 2: Replace All SAS Functions with the Oracle Equivalent

In this example, the equivalent Oracle functions in a SAS data set replace all SAS functions that contain the letter I.

```

proc sql;
create table work.replace as select *
  from work.allfuncs where sasfuncname like '%I%';
quit;

libname mydblib oracle sql_functions="EXTERNAL_REPLACE=work.replace"
  sql_functions_copy=saslog;

```

## Example 3: Add a New Function

```

data work.newfunc;
SASFUNCNAME = "sasname";
SASFUNCNAMELEN = 7;
DBMSFUNCNAME = "DBMSUDFName";
DBMSFUNCNAMELEN = 11;
FUNCTION_CATEGORY = "CONSTANT";
FUNC_USAGE_CONTEXT = "WHERE_ORDERBY";
FUNCTION_RETURNTYPE = "NUMERIC";
FUNCTION_NUM_ARGS = 0;
CONVERT_ARGS = 0;
ENGINEINDEX = 0;
output;
run;

/* Add function to existing in-memory function list */
libname mydblib oracle sql_functions="EXTERNAL_APPEND=work.newfunc"
  sql_functions_copy=saslog;

```

## Example 4: Add and Run the Netezza Variant of the SOUNDEX Function

Netezza supports the NYSIIS function, which is a variant of the SOUNDEX function and produces similar results. This example shows how to specify that NYSIIS should be called when the SOUNDEX function is passed to a Netezza DBMS.

```

data work.newfunc;
SASFUNCNAME = "SOUNDEX";
SASFUNCNAMELEN = 7;
DBMSFUNCNAME = "NYSIIS";
DBMSFUNCNAMELEN = 6;

```

```

FUNCTION_CATEGORY = "SCALAR";
FUNC_USAGE_CONTEXT = "SELECT_LIST";
FUNCTION_RETURNTYPE = "CHAR";
FUNCTION_NUM_ARGS = 1;
CONVERT_ARGS = 0;
ENGINEINDEX = 0;
output;
run;

options sastrace=',,d' sastraceloc=saslog nostsuffix;

/* Add function to existing in-memory function list */
libname x netezza server=&server. database=&database. uid=&user.
pwd=&password.
sql_functions="EXTERNAL_APPEND=work.newfunc"
sql_functions_copy=saslog;

proc sql;
select distinct soundex('oliver') from x._v_dual;
quit;

/* query returns */
/*
NYSIIS
-----
OLAVAR
*/

```

## SQL\_FUNCTIONS\_COPY= LIBNAME Statement Option

Writes the function associated with this particular LIBNAME statement to a SAS data set or the SAS log.

|               |                                                                                                                                                                                                                                                                                                                                                           |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                              |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                          |
| Default:      | none                                                                                                                                                                                                                                                                                                                                                      |
| Restrictions: | You must specify a two-part data set name, such as < <i>libref.member</i> > or an error results. < <i>libref.member</i> > must be a SAS data set. It is not checked to make sure that it is assigned to the default Base SAS engine.                                                                                                                      |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                  |
| Notes:        | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Hadoop, JDBC, and Microsoft SQL Server was added in SAS Viya 3.4.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Spark was added in SAS 9.4M7. |

Support for Yellowbrick was added in SAS 9.4M7.

See: [SQL\\_FUNCTIONS= LIBNAME option](#)

## Syntax

**SQL\_FUNCTIONS\_COPY=***libref.member* | SASLOG

### Syntax Description

***libref.member***

writes the current in-memory function list to a user-specified SAS data set for this particular LIBNAME statement.

**SASLOG**

writes the current in-memory function list to the SAS log for this particular LIBNAME statement.

## SQL\_OJ\_ANSI= LIBNAME Statement Option

Specifies whether to pass ANSI outer-join syntax through to the database.

Valid in: SAS/ACCESS LIBNAME statement

Defaults: Data Set Control

NO

Restriction: SAP ASE can process SQL outer joins only if the version of the Adaptive Server Enterprise (ASE) database is 12.5.2 or higher.

Data source: SAP ASE

## Syntax

**SQL\_OJ\_ANSI=**YES | NO

### Syntax Description

**YES**

specifies that ANSI outer-join syntax is passed through to the database.

**NO**

disables pass-through of ANSI outer-joins.

## SQLGENERATION= LIBNAME Statement Option

Specifies whether and when SAS procedures generate SQL for in-database processing of source data.

|               |                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                               |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                                                           |
| Default:      | none                                                                                                                                                                                                                                                                                                                                                                                       |
| Restrictions: | A value of NONE or DBMS modifies only the primary state that is specified on the system option.<br>If you are using the Metadata LIBNAME Engine, the only valid SQLGENERATION= modifiers are NONE and DBMS. The engine ignores the DBMS=, EXCLUDEDB=, and EXCLUDEPROC= modifiers.                                                                                                          |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, Netezza, Oracle, PostgreSQL, SAP HANA, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                                                                                                 |
| Notes:        | Support for HAWQ was added in SAS 9.4M3.<br>Support for PostgreSQL was added in the April 2016 release.<br>Support for JDBC was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Google BigQuery was added in SAS 9.4M7.<br>Support for Spark was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:          | <a href="#">SQLGENERATION= system option</a> , <a href="#">Precedence of Values for SQLGENERATION= LIBNAME and System Options</a> , <a href="#">SAS In-Database Products: Administrator's Guide</a>                                                                                                                                                                                        |

---

## Syntax

**SQLGENERATION=NONE | DBMS**

### Required Arguments

#### NONE

prevents those SAS procedures that are enabled for in-database processing from generating SQL for in-database processing.

#### DBMS

allows those SAS procedures that are enabled for in-database processing to generate SQL for in-database processing.

---

## SSL\_CA= LIBNAME Statement Option

Specifies the full path name to the certificate authority file.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Access

Default: none

Restriction: This option is not supported on AIX or HP-UX platforms.

Data source: MySQL

Note: Support for this LIBNAME option was added in SAS 9.4M4.

See: [SSL\\_CERT= LIBNAME option](#), [SSL\\_CIPHER= LIBNAME option](#), [SSL\\_KEY= LIBNAME option](#)

Example:

```
libname db mysql user=myuser database=myDB server=myServer port=3306  
      password=myPwd dbconinit="SET names utf8" PRESERVE_COL_NAMES=YES  
      PRESERVE_TAB_NAMES=YES  
      ssl_key="/u/myuser/mysql/newcerts/client-key.pem"  
      ssl_cert="/u/myuser/mysql/newcerts/client-cert.pem"  
      ssl_ca="/u/myuser/mysql/newcerts/ca-cert.pem"  
      ssl_cipher="DHE-RSA-AES256-SHA" ;
```

---

## Syntax

**SSL\_CA="path-and-file-name"**

### Syntax Description

#### ***path-and-file-name***

specifies the full path, including file name, to the certificate authority (CA) file to use to establish a secure connection to the MySQL server.

---

## Details

Certificate files enable MySQL to support secure connections using TLS encryption and validation. A CA file contains a list of trusted certificate authorities. A CA file is configured as part of configuring your MySQL server to use secure connections. The CA file must be specified in order to establish a secure connection to the server. For more information about configuring MySQL and creating CA files, see the security information in your MySQL Server documentation.

---

## SSL\_CERT= LIBNAME Statement Option

Specifies the full path name to the certificate file.

|              |                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                  |
| Category:    | Data Access                                                                                                                   |
| Default:     | none                                                                                                                          |
| Restriction: | This option is not supported on AIX or HP-UX platforms.                                                                       |
| Data source: | MySQL                                                                                                                         |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M4.                                                                       |
| See:         | <a href="#">SSL_CA= LIBNAME option</a> , <a href="#">SSL_CIPHER= LIBNAME option</a> , <a href="#">SSL_KEY= LIBNAME option</a> |

Example:

```
libname db mysql user=myuser database=myDB server=myServer port=3306
          password=myPwd dbconinit="SET names utf8" PRESERVE_COL_NAMES=YES
          PRESERVE_TAB_NAMES=YES
          ssl_key="/u/myuser/mysql/newcerts/client-key.pem"
          ssl_cert="/u/myuser/mysql/newcerts/client-cert.pem"
          ssl_ca="/u/myuser/mysql/newcerts/ca-cert.pem"
          ssl_cipher="DHE-RSA-AES256-SHA" ;
```

---

## Syntax

**SSL\_CERT="*path-name*"**

### Syntax Description

***path-name***

specifies the full path name to the certificate file to use to establish a secure connection to the MySQL server.

---

## Details

Certificate files enable MySQL to support secure connections using TLS encryption and validation. A certificate file identifies the server public key certificate. A certificate file is configured as part of configuring your MySQL server to use secure connections. The certificate file must be specified in order to establish a secure connection to the server. For more information about configuring MySQL and creating certificates, see the security information in your MySQL Server documentation.

---

## SSL\_CIPHER= LIBNAME Statement Option

Specifies a list of permissible ciphers to use for TLS encryption.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Access

Default: none

Restriction: This option is not supported on AIX or HP-UX platforms.

Data source: MySQL

Note: Support for this LIBNAME option was added in SAS 9.4M4.

See: [SSL\\_CA= LIBNAME option](#), [SSL\\_CERT= LIBNAME option](#), [SSL\\_KEY= LIBNAME option](#)

Example:

```
libname db mysql user=myuser database=myDB server=myServer port=3306
          password=myPwd dbconinit="SET names utf8" PRESERVE_COL_NAMES=YES
          PRESERVE_TAB_NAMES=YES
          ssl_key="/u/myuser/mysql/newcerts/client-key.pem"
          ssl_cert="/u/myuser/mysql/newcerts/client-cert.pem"
          ssl_ca="/u/myuser/mysql/newcerts/ca-cert.pem"
          ssl_cipher="DHE-RSA-AES256-SHA" ;
```

---

## Syntax

**SSL\_CIPHER="***cipher-list***"**

### Syntax Description

#### *cipher-list*

specifies a list of ciphers that are permitted to be used for TLS encryption.

---

## Details

Each secure connection to the MySQL Server uses an encryption cipher. The SSL\_CIPHER= option provides a list of ciphers that are permitted on any particular secure connection. For more information about using MySQL Server ciphers, see the security information in your MySQL Server documentation.

---

## SSL\_KEY= LIBNAME Statement Option

Specifies the full path name to the key file.

|              |                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                      |
| Category:    | Data Access                                                                                                                                                                                                                                                                                                                                                                                       |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                                                              |
| Restriction: | This option is not supported on AIX or HP-UX platforms.                                                                                                                                                                                                                                                                                                                                           |
| Data source: | MySQL                                                                                                                                                                                                                                                                                                                                                                                             |
| Note:        | Support for this LIBNAME option was added in SAS 9.4M4.                                                                                                                                                                                                                                                                                                                                           |
| See:         | <a href="#">SSL_CA= LIBNAME option</a> , <a href="#">SSL_CERT= LIBNAME option</a> , <a href="#">SSL_CIPHER= LIBNAME option</a>                                                                                                                                                                                                                                                                    |
| Example:     | <pre>libname db mysql user=myuser database=myDB server=myServer port=3306       password=myPwd dbconinit="SET names utf8" PRESERVE_COL_NAMES=YES       PRESERVE_TAB_NAMES=YES       ssl_key="/u/myuser/mysql/newcerts/client-key.pem"       ssl_cert="/u/myuser/mysql/newcerts/client-cert.pem"       ssl_ca="/u/myuser/mysql/newcerts/ca-cert.pem"       ssl_cipher="DHE-RSA-AES256-SHA" ;</pre> |

---

## Syntax

**SSL\_KEY="path"**

### Syntax Description

***path***

specifies the full path name to the key file to use to establish a secure connection to the MySQL server.

---

## Details

A key file is configured as part of configuring your MySQL server. The key file must be specified in order to establish a secure connection to the server. For more information about creating an SSL key file, see your MySQL Server documentation.

---

## SSLMODE= LIBNAME Statement Option

Specifies support for a TLS encrypted connection to your data source.

|              |                                                          |
|--------------|----------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                             |
| Category:    | Data Access                                              |
| Default:     | prefer                                                   |
| Restriction: | You must specify the value for this option in lowercase. |
| Data source: | PostgreSQL, Yellowbrick                                  |

Notes: Support for this option was added in SAS Viya 3.4.  
Support for Yellowbrick was added in SAS 9.4M7.

## Syntax

**SSLMODE='value'**

### Syntax Description

#### **value**

specifies whether to support encryption when SAS/ACCESS connects to your database.

Here are the possible values:

|             |                                                                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| allow       | Encryption is not required, but encryption is accepted if the DBMS server requires it.                                                                          |
| disable     | Do not use encryption when connecting to the database.                                                                                                          |
| prefer      | Encryption is not required, but encryption should be used if it is available.                                                                                   |
| require     | Encryption is required when connecting to the database.                                                                                                         |
| verify-ca   | Encryption is required, and an authentication certificate might be required. Whether an authentication certificate is required depends on your database policy. |
| verify-full | Encryption and an authentication certificate are required.                                                                                                      |

## Details

In order for a database connection to be secure, encryption usage must be configured on the DBMS server and on the SAS/ACCESS client.

The following table shows whether third-party observation of connection parameters and data is possible for each SSLMODE= value. It also shows whether there is protection against man-in-the-middle (MITM) attacks, in which data is modified or diverted during transfer even if the data is encrypted. The value for SSLMODE= specifies whether to use certificate verification to prevent these attacks.

**Table 12.9** Security Details for SSLMODE= Values

| SSLMODE= Value | Protects from Third-Party Observation?        | Protects from MITM Attacks? |
|----------------|-----------------------------------------------|-----------------------------|
| allow          | Yes, if encryption is used.<br>Otherwise, no. | No                          |
| disable        | No                                            | No                          |

| SSLMODE= Value | Protects from Third-Party Observation?        | Protects from MITM Attacks?                              |
|----------------|-----------------------------------------------|----------------------------------------------------------|
| prefer         | Yes, if encryption is used.<br>Otherwise, no. | No                                                       |
| require        | Yes                                           | No                                                       |
| verify-ca      | Yes                                           | Depends on certificate authentication policy of the DBMS |
| verify-full    | Yes                                           | Yes                                                      |

## STRINGDATES= LIBNAME Statement Option

Specifies whether to read date and time values from the database as character strings or as numeric date values.

- Valid in: SAS/ACCESS LIBNAME statement, CONNECT statement
- Category: Data Set Control
- Alias: STRDATES= [Greenplum, HAWQ, Microsoft SQL Server, PostgreSQL, SAP IQ]
- Default: NO
- Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, ODBC, OLE DB, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick
- Notes: Support for HAWQ was added in SAS 9.4M3.  
Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.  
Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.  
Support for Yellowbrick was added in SAS 9.4M7.
- Tip: Use STRINGDATES=NO for SAS 6 compatibility.

## Syntax

**STRINGDATES=YES | NO**

### Syntax Description

#### YES

specifies that SAS reads date and time values as character strings.

#### NO

specifies that SAS reads date and time values as numeric date values.

---

## SUB\_CHAR= LIBNAME Statement Option

Specifies a substitution character to use in place of any invalid characters that cannot be represented in the SAS session encoding.

|              |                                                                                                                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                            |
| Category:    | Data Set Control                                                                                                                                                                                        |
| Aliases:     | SUBCHAR=<br>SUBSTITUTION_CHAR=                                                                                                                                                                          |
| Default:     | none                                                                                                                                                                                                    |
| Interaction: | This option can be used with the BULKUNLOAD= LIBNAME option.                                                                                                                                            |
| Data source: | Amazon Redshift, Greenplum, Hadoop, HAWQ, Impala, JDBC, Netezza, PostgreSQL, SAP HANA, Spark, Vertica                                                                                                   |
| Notes:       | Support for this option was added in SAS 9.4M6.<br>Support for Greenplum, HAWQ, Impala, SAP HANA, and Vertica was added in SAS Viya 3.5.<br>Support for Hadoop, JDBC, and Spark was added in SAS 9.4M7. |
| See:         | <a href="#">BULKUNLOAD= LIBNAME option</a>                                                                                                                                                              |

---

## Syntax

**SUB\_CHAR=***value*

### Syntax Description

#### ***value***

specifies the substitution character to use in place of a character that cannot be represented in the SAS session encoding.

Here are the possible values for this option:

QUESTIONMARK specifies that a question mark ('?') replaces an invalid character during transcoding.

SPACE specifies that a space replaces an invalid character during transcoding.

SUB specifies that the default substitution character for the source encoding replaces an invalid character during transcoding. The default substitution character varies depending on your source encoding and on your system platform. For example, the substitution character might be represented by a '0x1A' character on Windows and by a '?' on Linux machines.

UESC specifies that the 'uddd' value replaces an invalid character during transcoding. This UESC value is

typically used only during debugging and is not intended to be used for large-scale jobs.

---

## Details

This option applies only when SAS/ACCESS transcodes the data into the SAS session encoding when retrieving data into SAS. If your data is transcoded by your DBMS client to the SAS session encoding before loading data into SAS, then this option is not used.

The default SAS session encoding is UTF-8. If your data must be transcoded from another encoding to the SAS session encoding and if your data contains characters that cannot be transcoded to the destination encoding, then warnings are generated in the SAS log.

To see details about the rows and columns that are affected by transcoding errors, specify the SASTRACE= option:

```
options sastrace=',,d';
```

---

## SYNONYMS= LIBNAME Statement Option

Specifies whether PROC DATASETS shows synonyms, tables, views, or materialized views for the current user and schema if you specified the SCHEMA= option.

|              |                                                                                  |
|--------------|----------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                     |
| Category:    | Data Set Control                                                                 |
| Alias:       | SHOW_SYNONYMS=                                                                   |
| Default:     | YES                                                                              |
| Data source: | Netezza                                                                          |
| Note:        | Support for this LIBNAME option was added for SAS 9.4.                           |
| See:         | <a href="#">SCHEMA= LIBNAME option</a> , <a href="#">SCHEMA= data set option</a> |

---

## Syntax

**SYNONYMS=YES | NO**

### Syntax Description

#### YES

specifies that PROC DATASETS shows only synonyms that represent tables, views, or materialized views for the current user.

#### NO

specifies that PROC DATASETS shows only tables, views, or materialized views for the current user.

## Details

Rather than submit PROC DATASETS, you can select the libref in SAS Explorer to obtain this same information. By default, no PUBLIC synonyms are displayed unless you specify SCHEMA=PUBLIC.

When you specify only the **SCHEMA=** option, the current schema is always displayed with the appropriate privileges.

Synonyms, tables, views, and materialized views in a different schema are also displayed.

---

## TABLE\_TYPE= LIBNAME Statement Option

Specifies the type of temporary tables and table storage when the engine creates tables in SAP HANA.

|                            |                                                                                                                                                                                                                                                                                                          |                |                   |                  |                   |                            |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-------------------|------------------|-------------------|----------------------------|
| Valid in:                  | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                             |                |                   |                  |                   |                            |
| Category:                  | Data Set Control                                                                                                                                                                                                                                                                                         |                |                   |                  |                   |                            |
| Default:                   | none                                                                                                                                                                                                                                                                                                     |                |                   |                  |                   |                            |
| Restrictions:              | <p>If you do not specify a value for this option, tables that are created in SAP HANA follow the SAP HANA default for row or column store.</p> <p>ROW and COLUMN are mutually exclusive.</p> <p>LOCAL and GLOBAL are mutually exclusive.</p> <p>Do not specify LOCAL or GLOBAL for permanent tables.</p> |                |                   |                  |                   |                            |
| Interaction:               | The TABLE_TYPE data set option overrides the TABLE_TYPE LIBNAME option.                                                                                                                                                                                                                                  |                |                   |                  |                   |                            |
| Data source:               | SAP HANA                                                                                                                                                                                                                                                                                                 |                |                   |                  |                   |                            |
| See:                       | <a href="#">CONNECTION=</a> (for sharing sessions across librefs), <a href="#">TABLE_TYPE=</a> data set option                                                                                                                                                                                           |                |                   |                  |                   |                            |
| Examples:                  | <table> <tr> <td>TABLE_TYPE=ROW</td> </tr> <tr> <td>TABLE_TYPE=COLUMN</td> </tr> <tr> <td>TABLE_TYPE=LOCAL</td> </tr> <tr> <td>TABLE_TYPE=GLOBAL</td> </tr> <tr> <td>TABLE_TYPE=(COLUMN GLOBAL)</td> </tr> </table>                                                                                      | TABLE_TYPE=ROW | TABLE_TYPE=COLUMN | TABLE_TYPE=LOCAL | TABLE_TYPE=GLOBAL | TABLE_TYPE=(COLUMN GLOBAL) |
| TABLE_TYPE=ROW             |                                                                                                                                                                                                                                                                                                          |                |                   |                  |                   |                            |
| TABLE_TYPE=COLUMN          |                                                                                                                                                                                                                                                                                                          |                |                   |                  |                   |                            |
| TABLE_TYPE=LOCAL           |                                                                                                                                                                                                                                                                                                          |                |                   |                  |                   |                            |
| TABLE_TYPE=GLOBAL          |                                                                                                                                                                                                                                                                                                          |                |                   |                  |                   |                            |
| TABLE_TYPE=(COLUMN GLOBAL) |                                                                                                                                                                                                                                                                                                          |                |                   |                  |                   |                            |

---

## Syntax

**TABLE\_TYPE=ROW | COLUMN**  
**TABLE\_TYPE=LOCAL | GLOBAL**  
**TABLE\_TYPE=(COLUMN LOCAL)**

### Syntax Description

#### **ROW**

creates a table using ROW-based storage in SAP HANA.

**COLUMN**

creates a table using COLUMN-based storage in SAP HANA.

**LOCAL**

creates a local temporary table in SAP HANA. The table definition and data are visible only in the current session.

Alias: LOCAL TEMPORARY

**GLOBAL**

creates a global temporary table in SAP HANA. The global temporary tables are globally available, and the data is visible only in the current session.

Alias: GLOBAL TEMPORARY

## Details

This option takes effect when a SAS program creates a table in SAP HANA.

When more than one argument is specified, place the arguments within parentheses () and separate them with a space.

## TEMP\_CTAS= LIBNAME Statement Option

Specifies whether to create an intermediate (temporary) table in Hive.

|              |                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                      |
| Category:    | Data Set Control                                                                                  |
| Default:     | YES [when the Hive property <code>hive.server2.enable.doAs</code> is set to TRUE. See “Details.”] |
| Interaction: | works with the READ_METHOD= LIBNAME option                                                        |
| Data source: | Hadoop                                                                                            |
| Note:        | Support for this option was added in SAS 9.4M6.                                                   |
| See:         | <a href="#">READ_METHOD= LIBNAME option</a>                                                       |

## Syntax

**TEMP\_CTAS=YES | NO**

## Syntax Description

**YES**

creates an intermediate (temporary) table in Hive.

**NO**

does not create an intermediate (temporary) table in Hive.

## Details

When READ\_METHOD=HDFS, SAS/ACCESS Interface to Hadoop creates an intermediate result set to HDFS. This set can be created as a TEMPORARY table in Hive or as an EXTERNAL table in HDFS. In each case, the intermediate result set is deleted when the query operation completes. Creating the result set as a TEMPORARY table offers the advantage of Hive managing the life cycle of the table.

The default for TEMP\_CTAS is YES when the Hive property `hive.server2.enable.doAs` is set to TRUE. In this case, the Hive process impersonates the current user to create the intermediate result set with the current user's permissions. This works even if the intermediate result set is created as a Hive TEMPORARY table.

When `hive.server2.enable.doAs` is set to FALSE, the Hive process runs under the "hive" user ID. In this case, the intermediate result set must be created as an external table so that SAS/ACCESS to Hadoop can read the table. The reason for that is that the running user will not have permission to read a temporary table that the "hive" user owns.

The user typically does not have to set this option. It is provided for cases when the intermediate result set is created as a TEMPORARY table and cannot be read.

You can control this behavior by setting the environment variable `SAS_HADOOP_TEMP_CTAS=YES/NO`.

---

## TEMPORAL\_QUALIFIER= LIBNAME Statement Option

Specifies time-dimension criteria for retrieving data from Teradata.

|              |                                                                                            |
|--------------|--------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                               |
| Category:    | Data Set Control                                                                           |
| Defaults:    | CURRENT VALIDTIME (valid-time column)<br>CURRENT TRANSACTIONTIME (transaction-time column) |
| Interaction: | Specifying values in a DATA step overrides LIBNAME values.                                 |
| Data source: | Teradata                                                                                   |
| See:         | <a href="#">TEMPORAL_QUALIFIER= data set option</a>                                        |

---

## Syntax

```

TEMPORAL_QUALIFIER=CURRENT VALIDTIME
| VALIDTIME AS OF PERIOD 'period'
| SEQUENCED VALIDTIME PERIOD 'period'
| NONSEQUENCED VALIDTIME PERIOD 'period'

TEMPORAL_QUALIFIER=CURRENT TRANSACTIONTIME
| TRANSACTIONTIME AS OF 'period'
```

## | NONSEQUENCED TRANSACTIONTIME

### Syntax Description

#### **CURRENT VALIDTIME**

selects rows that are valid at the current time.

#### **VALIDTIME AS OF PERIOD '*period*'**

selects rows with valid-time periods that overlap the specified AS OF period. For the period, you can specify either a single date or a time period (date range) by specifying a start date and an end date.

#### **SEQUENCED VALIDTIME PERIOD '*period*'**

selects history, current, or future rows that are valid for the specified time period.

#### **NONSEQUENCED VALIDTIME PERIOD '*period*'**

treats the table as nontemporal.

#### **CURRENT TRANSACTIONTIME**

selects rows that are open in transaction time.

#### **TRANSACTIONTIME AS OF '*period*'**

selects rows with transaction-time periods that overlap the specified AS OF period. For the period, you can specify either a single date or a time period (date range) by specifying a start date and an end date.

#### **NONSEQUENCED TRANSACTIONTIME**

treats the table as nontemporal.

## Details

Use temporal qualifiers to specify time criteria for selecting data from temporal tables.

Temporal qualifiers that you specify in a LIBNAME statement apply only to that Teradata session and are implemented through session commands that are issued at connect time. For example, if you specify TEMPORAL\_QUALIFIER='AS OF PERIOD '(1999-01-01, 2099-01-05)' in a LIBNAME statement, below is the Teradata SET SESSION command that is issued at connect time. The SQL is submitted as usual.

```
.SET SESSION ASOF PERIOD '(1999-01-01, 2099-01-05)'
```

## Example: Select ValidTime Rows for a Specific Date for the Current Session

In this example, valid-time rows are selected for a specific date from the Mytest data set.

```
/* Consider data as of 1995-01-01. */
libname x teradata user=myusr1 pw=mypwd1 server=mysrv1
TEMPORAL_QUALIFIER='VALIDTIME AS OF DATE '1995-01-01' '
```

```
/* .SET SESSION VALIDTIME ASOF DATE'1995-01-01'; is issued
   before submitting the SQL "Select * from mytest" */
proc print data=x.mytest (DBSLICEPARM=ALL) ;
run;
```

---

## TENACITY= LIBNAME Statement Option

Specifies how many hours that FastExport, FastLoad, or MultiLoad continues to try to log on again to Teradata if the maximum number of Teradata utilities are already running.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Defaults:    | 0 (FastLoad)<br>4 (FastExport, MultiLoad)                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Tip:         | The data set option has precedence over the LIBNAME option.                                                                                                                                                                                                                                                                                                                                                                                               |
| See:         | <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">FASTEXPORT= LIBNAME option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">SLEEP= LIBNAME option</a> , <a href="#">SLEEP= data set option</a> , <a href="#">TENACITY= data set option</a> , <a href="#">Using the TPT API</a> |

---

## Syntax

**TENACITY=***number-of-hours*

### Syntax Description

***number-of-hours***

specifies the number of hours to continue to try again to log on to Teradata.

---

## Details

Use this option to indicate to [FastExport](#), [FastLoad](#), or [MultiLoad](#) how long to continue retrying a logon operation when the maximum number of utilities are already running. (The maximum number of Teradata utilities that can run concurrently varies from 5 to 15, depending on the database server value.) The default value for TENACITY= is 4 hours. The value specified for TENACITY= must be greater than zero.

Use TENACITY= with [SLEEP=](#). SLEEP= specifies the number of minutes that FastExport, FastLoad, or MultiLoad waits before it retries logging on to Teradata. SLEEP= and TENACITY= function very much like the SLEEP and TENACITY run-time options of the native Teradata FastExport, FastLoad, or MultiLoad utility.

Here is an example of the message that is written to the SAS log if the time period that TENACITY= specifies is exceeded.

ERROR: MultiLoad failed unexpectedly with returncode 12

Check the FastExport, FastLoad, or MultiLoad log for more information about the cause of the FastExport, FastLoad, or MultiLoad failure. SAS does not receive any informational messages from Teradata in either of these situations:

- when the currently run FastExport, FastLoad, or MultiLoad process waits because the maximum number of utilities are already running
- if FastExport, FastLoad, or MultiLoad is terminated because the time limit that TENACITY= specifies has been exceeded

The native Teradata FastExport, FastLoad, or MultiLoad utility sends messages associated with SLEEP= and TENACITY= only to the FastExport, FastLoad, or MultiLoad log. Therefore, nothing is written to the SAS log.

## TPT= LIBNAME Statement Option

Specifies whether SAS uses the Teradata Parallel Transporter (TPT) API to load data when SAS requests a Fastload, MultiLoad, or Multi-Statement insert.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: YES

Data source: Teradata

See: [BULKLOAD= LIBNAME option](#), [BULKLOAD= data set option](#), [LOGDB= LIBNAME option](#), [MULTILOAD= data set option](#), [MULTISTMT= data set option](#), [TPT= data set option](#), [TPT\\_APPL\\_PHASE= data set option](#), [TPT\\_BUFFER\\_SIZE= data set option](#), [TPT\\_CHECKPOINT= data set option](#), [TPT\\_DATA\\_ENCRYPTION= data set option](#), [TPT\\_ERROR\\_TABLE\\_1= data set option](#), [TPT\\_ERROR\\_TABLE\\_2= data set option](#), [TPT\\_LOG\\_TABLE= data set option](#), [TPT\\_MAX\\_SESSIONS= LIBNAME option](#), [TPT\\_MAX\\_SESSIONS= data set option](#), [TPT\\_MIN\\_SESSIONS= data set option](#), [TPT\\_PACK= data set option](#), [TPT\\_PACKMAXIMUM= data set option](#), [TPT\\_RESTART= data set option](#), [TPT\\_TRACE\\_LEVEL= data set option](#), [TPT\\_TRACE\\_LEVEL\\_INF= data set option](#), [TPT\\_TRACE\\_OUTPUT= data set option](#), [TPT\\_WORK\\_TABLE= data set option](#), "Maximizing Teradata Load and Read Performance", "Using the TPT API"

## Syntax

**TPT=YES | NO**

## Syntax Description

### YES

specifies that SAS uses the TPT API when Fastload, MultiLoad, or Multi-Statement insert is requested.

**NO**

specifies that SAS does not use the TPT API when Fastload, MultiLoad, or Multi-Statement insert is requested.

---

## Details

By using the TPT API, you can load data into a Teradata table without working directly with such stand-alone Teradata utilities as [FastLoad](#), [MultiLoad](#), or TPump. When TPT=YES (the default), SAS uses the TPT API load driver for FastLoad, the update driver for MultiLoad, the TPT Export driver for FastExport, and the stream driver for Multi-Statement insert.

When TPT=YES, sometimes SAS cannot use the TPT API due to an error or because it is not installed on the system. When this happens, SAS does not produce an error, but it still tries to load data using the requested load method (Fastload, MultiLoad, or Multi-Statement insert). To check whether SAS used the TPT API to load data, look for a similar message to this one in the SAS log.

NOTE: Teradata connection: TPT FastLoad/MultiLoad/MultiStatement  
insert has red n row(s).

---

## Example: Load SAS Data into Teradata

In this example, SAS data is loaded into Teradata using the TPT API. This is the default method of loading when Fastload, MultiLoad, or Multi-Statement insert are requested. SAS still tries to load data even if it cannot use the TPT API.

```
libname tera teradata user=myusr1 pw=mypwd1 TPT=YES;
/* Create data */
data testdata;
do i=1 to 100;
  output;
end;
run;

/* Load using MultiLoad TPT.  This note appears in the SAS
log if SAS uses TPT.  NOTE: Teradata connection:
TPT MultiLoad has inserted 100 row(s).*/
data tera.testdata(MULTILOAD=YES);
set testdata;
run;
```

---

## TPT\_DATA\_ENCRYPTION= LIBNAME Statement Option

Specifies whether to encrypt SQL requests, responses, and data when using the Teradata Parallel Transport (TPT) API.

|              |                                                 |
|--------------|-------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                    |
| Categories:  | Bulk Loading<br>Data Set Control                |
| Default:     | NO                                              |
| Requirement: | To use this option, specify TPT=YES.            |
| Data source: | Teradata                                        |
| Note:        | Support for this option was added in SAS 9.4M7. |
| See:         | <a href="#">TPT= LIBNAME option</a>             |

## Syntax

**TPT\_DATA\_ENCRYPTION=YES | NO**

### Required Argument

#### YES | NO

specifies whether to encrypt SQL requests, responses, and data when using the TPT API.

## Details

This option pertains to LOAD, UPDATE, EXPORT, or STREAM operations that use the TPT API.

## TPT\_MAX\_SESSIONS= LIBNAME Statement Option

Specifies the maximum number of sessions for Teradata to use when using the TPT API with FastLoad, MultiLoad, or Multi-Statement insert.

|              |                                                                                                                                                            |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                               |
| Category:    | Data Set Control                                                                                                                                           |
| Default:     | 4                                                                                                                                                          |
| Restriction: | This option is valid only when using the TPT API.                                                                                                          |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                        |
| Interaction: | [precedence] 1. TPT_MAX_SESSIONS= data set option, 2. TPT_MAX_SESSIONS= LIBNAME option, 3. SAS_TPT_MAX_SESSIONS environment variable, 4. the default value |
| Data source: | Teradata                                                                                                                                                   |
| Note:        | Support for this LIBNAME option was added for SAS 9.4.                                                                                                     |

See: [BULKLOAD= LIBNAME option](#), [BULKLOAD= data set option](#), [Maximizing Teradata Load Performance](#), [MULTILOAD= data set option](#), [MULTISTMT= data set option](#), [TPT= LIBNAME option](#), [TPT= data set option](#), [TPT\\_MAX\\_SESSIONS= data set option](#), [TPT\\_MIN\\_SESSIONS= data set option](#)[Using the TPT API](#)

Examples: Set the SAS\_TPT\_MAX\_SESSIONS environment variable on PC hosts:

```
SAS_TPT_MAX_SESSIONS 4
```

Set the SAS\_TPT\_MAX\_SESSIONS environment variable on UNIX hosts:

```
export SAS_TPT_MAX_SESSIONS=4
```

Set the SAS\_TPT\_MAX\_SESSION environment variable at SAS invocation:

```
sas -set SAS_TPT_MAX_SESSION 4
```

## Syntax

**TPT\_MAX\_SESSIONS=***integer*

### Syntax Description

#### *integer*

specifies the maximum number of sessions for Teradata to use when using the TPT API to load data with [FastLoad](#), [MultiLoad](#), or [Multi-Statement](#) insert.

## Details

You can control the number of sessions for Teradata to use when using the TPT API to load data with MultiLoad. The maximum value cannot be more than the number of available Access Module Processors (AMPs). See your Teradata documentation for details.

## TPT\_MIN\_SESSIONS= LIBNAME Statement Option

Specifies the minimum number of sessions for Teradata to use when using the TPT API with FastLoad, MultiLoad, or Multi-Statement insert.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: 1

Restriction: This option is valid only when using the TPT API.

Requirement: To use this option, you must first specify TPT=YES.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Interaction: | [precedence] 1. TPT_MIN_SESSIONS= data set option, 2. TPT_MIN_SESSIONS= LIBNAME option, 3. SAS_TPT_MIN_SESSIONS environment variable, 4. the default value                                                                                                                                                                                                                                                                                                                                                      |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Note:        | Support for this LIBNAME option was added for SAS 9.4.                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">MULTISTMT= data set option</a> , <a href="#">TPT= LIBNAME option</a> , <a href="#">TPT= data set option</a> , <a href="#">TPT_MAX_SESSIONS= LIBNAME option</a> , <a href="#">TPT_MAX_SESSIONS= data set option</a> , <a href="#">TPT_MIN_SESSIONS= data set option</a> Using the <a href="#">TPT API</a> |
| Examples:    | <p>Set the SAS_TPT_MIN_SESSIONS environment variable on PC hosts:</p> <pre>SAS_TPT_MIN_SESSIONS=1</pre> <p>Set the SAS_TPT_MIN_SESSIONS environment variable on UNIX hosts:</p> <pre>export SAS_TPT_MIN_SESSIONS=1</pre> <p>Set the SAS_TPT_MIN_SESSIONS environment variable at SAS invocation:</p> <pre>sas -set SAS_TPT_MIN_SESSIONS 1</pre>                                                                                                                                                                 |

---

## Syntax

**TPT\_MIN\_SESSIONS=*integer***

### Syntax Description

***integer***

specifies the minimum number of sessions for Teradata to use when using the TPT API to load data with [FastLoad](#), [MultiLoad](#), or [Multi-Statement insert](#).

---

## Details

You can control the number of sessions for Teradata to use when using the TPT API to load data with MultiLoad. The minimum value cannot be more than the value for the TPT\_MAX\_SESSIONS= LIBNAME option.

---

## TPT\_UNICODE\_PASSTHRU= LIBNAME Statement Option

Specifies whether to allow pass-through characters (PTC) to be loaded into SAS or written to Teradata.

|             |                                  |
|-------------|----------------------------------|
| Valid in:   | SAS/ACCESS LIBNAME statement     |
| Categories: | Bulk Loading<br>Data Set Control |

Default: NO

Requirement: To use this option, specify TPT=YES.

Data source: Teradata

Note: Support for this option was added in SAS 9.4M7.

See: [TPT= LIBNAME option](#)

## Syntax

**TPT\_UNICODE\_PASSTHRU=YES | NO**

### Required Argument

#### **YES | NO**

specifies whether to allow pass-through characters (PTC) to be imported into SAS or written to Teradata when using the Teradata Parallel Transport (TPT) API.

## Details

PTCs are the set of all possible Unicode characters and noncharacters, except for the 6.0 BMP characters, that are supported by Teradata.

This option pertains to LOAD, UPDATE, EXPORT, or STREAM operations that use the TPT API.

## TR\_ENABLE\_INTERRUPT= LIBNAME Statement Option

Allows interruption of any long-running SQL processes that are involved in creating the result set or spool file.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: NO

Restriction: Valid for only Windows and UNIX platforms

Data source: Teradata

## Syntax

**TR\_ENABLE\_INTERRUPT=YES | NO**

## Syntax Description

### YES

allows interruption of long-running SQL processes that are involved in creating the result set or spool file.

### NO

disables the interrupt processing code path.

## Details

When set to YES, here is how you can use this option:

- to interrupt any SELECT SQL statement that was submitted by using the SELECT \* FROM CONNECTION as a pass-through statement
- by using the **Interrupt/Attention** button to interrupt a query that you submitted through a DATA STEP, through PROC SQL implicit pass-through, or through explicit pass-through

Once the result set or spool file forms on the Teradata server and SAS is fetching the results, it is likely that the interrupt might no longer be available. You must wait until all results are fetched. The interrupt works in only one of these cases:

- when the Teradata server is building the result set or spool file
- if the Teradata server is in a wait state before building the result set or spool file because of locking

## Example

```
libname x teradata user=myusr1 pass=mypwd1
      TR_ENABLE_INTERRUPT=YES server=mysrv1;

      data _NULL_; set x.paul_test; run;

      proc datasets lib=x; quit;

      proc sql;
      create table work.a as select * from x.td_cancel, x.td_cancel;
      quit;

      proc sql; connect to teradata (user=myusr1 pass=mypwd1
      TR_ENABLE_INTERRUPT=YES server=mysrv1);
      select * from connection to teradata
      ( select * From td_cancel a , td_cancel b );
      quit;
```

## TRACE= LIBNAME Statement Option

Specifies whether to turn on tracing information for use in debugging.

|              |                                                                                                                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                                                                                   |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                  |
| Default:     | NO                                                                                                                                                                                                                                                                                                |
| Restriction: | PostgreSQL, Yellowbrick: This option is valid only on a Windows platform.                                                                                                                                                                                                                         |
| Data source: | Amazon Redshift, Aster, Google BigQuery, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, ODBC, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick                                                                                                                              |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">TRACEFILE= LIBNAME option</a>                                                                                                                                                                                                                                                         |

## Syntax

**TRACE=YES | NO**

### Syntax Description

#### YES

specifies that tracing is turned on, and the DBMS driver manager writes each function call to the trace file that TRACEFILE= specifies.

#### NO

specifies that tracing is not turned on.

## TRACEFILE= LIBNAME Statement Option

Specifies the file name to which the DBMS driver manager writes trace information.

|              |                                                                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                                                                                                                                                                            |
| Category:    | Data Set Control                                                                                                                                                                                                                           |
| Default:     | none                                                                                                                                                                                                                                       |
| Restriction: | PostgreSQL, Yellowbrick: This option is valid only on a Windows platform.                                                                                                                                                                  |
| Interaction: | TRACEFILE= is used only when TRACE=YES.                                                                                                                                                                                                    |
| Data source: | Amazon Redshift, Aster, Google BigQuery, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, ODBC, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick                                                                       |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> |

Support for Yellowbrick was added in SAS 9.4M7.

See: [TRACE= LIBNAME option](#)

## Syntax

**TRACEFILE=***file-name* | <>'*path-and-file-name*'>

## Details

If you specify a file name without a path, the SAS trace file is stored with your data files. If you specify a directory, enclose the fully qualified file name in single quotation marks.

If you do not specify the TRACEFILE= option, output is directed to a default file.

*Google BigQuery:* If you do not specify TRACEFILE=, output is directed to the ttrace.txt file.

## TRACEFLAGS= LIBNAME Statement Option

Specifies whether to control the properties of the trace file.

|              |                                                                                   |
|--------------|-----------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement                                   |
| Category:    | Data Set Control                                                                  |
| Default:     | none                                                                              |
| Data source: | Google BigQuery                                                                   |
| Note:        | Support for this option was added in the August 2019 release of SAS/ACCESS.       |
| See:         | <a href="#">TRACE= LIBNAME option</a> , <a href="#">TRACEFILE= LIBNAME option</a> |

## Syntax

**TRACEFLAGS=**[ALL](#) | [COLBINDINGS](#) | [GETDATA](#) | [PUTDATA](#) | [TIMESTAMP](#)

## Syntax Description

### **ALL**

enables tracing of all information that is available via this option.

### **COLBINDINGS**

enables tracing of column information for Read and Write operations.

### **GETDATA**

enables tracing of Read operations.

**PUTDATA**

enables tracing of Write operations.

**TIMESTAMP**

enables tracing of time-related data.

## TRANSCODE\_FAIL= LIBNAME Statement Option

Lets you specify how to handle processing and notification of transcoding errors.

|               |                                                                                                                                                                    |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | SAS/ACCESS LIBNAME statement                                                                                                                                       |
| Category:     | Data Set Control                                                                                                                                                   |
| Default:      | ERROR                                                                                                                                                              |
| Restrictions: | <p>This option is deprecated in SAS Viya 3.5. Use the SUB_CHAR= LIBNAME option instead.</p> <p>This option is not available for use with SAS Embedded Process.</p> |
| Data source:  | Hadoop, JDBC                                                                                                                                                       |
| Note:         | Support for JDBC was added in SAS Viya 3.4.                                                                                                                        |
| Tip:          | You can use TRANSCODE_FAIL= to determine whether you want to halt or continue processing when transcoding errors are encountered.                                  |
| See:          | <a href="#">SUB_CHAR= LIBNAME option</a> , <a href="#">SAS/ACCESS LIBNAME options for NLS</a> ,<br><a href="#">TRANSCODE_FAIL= data set option</a>                 |

## Syntax

**TRANSCODE\_FAIL=**[ERROR | WARNING | SILENT](#)

## Optional Arguments

**ERROR**

stops processing data and provides an informative error message.

**WARNING**

continues processing of data but provides an informative error message.

**SILENT**

continues processing of data but suppresses messages.

## Details

The TRANSCODE\_FAIL= LIBNAME option is deprecated in the SAS Viya 3.5 (November 2019) release. Using this option stops processing and results in an error in the SAS log. Use the SUB\_CHAR= LIBNAME option instead.

---

## UPDATE\_ISOLATION\_LEVEL= LIBNAME Statement Option

Specifies the degree of isolation of the current application process from other concurrently running application processes.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                                                                                                                                  |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                              |
| Alias:       | UIL= [Greenplum, HAWQ, SAP IQ]                                                                                                                                                                                                                                                                                                                                                                                                |
| Default:     | DBMS-specific                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Restriction: | DB2 under UNIX and PC Hosts, Greenplum, HAWQ, ODBC: If you do not specify UPDATE_LOCK_TYPE=ROW for these interfaces, this option is ignored.                                                                                                                                                                                                                                                                                  |
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP IQ, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                                                        |
| Notes:       | This option is ignored in the interfaces to DB2 under UNIX and PC Hosts and ODBC if you do not specify UPDATE_LOCK_TYPE=ROW.<br>Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                 |
| See:         | <a href="#">CONNECTION= LIBNAME option</a> , <a href="#">READ_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">READ_ISOLATION_LEVEL= data set option</a> , <a href="#">READ_LOCK_TYPE= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= data set option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= data set option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a> , <a href="#">UPDATE_LOCK_TYPE= data set option</a> |

---

## Syntax

**UPDATE\_ISOLATION\_LEVEL=***DBMS-specific-value*

### Syntax Description

---

## Details

The degree of isolation specifies the degree to which these items are affected:

- Rows that the current application reads and updates are available to other concurrently executing applications.
- Update activity of other concurrently executing application processes can affect the current application.

For more information, including a list of supported values, see the information about locking for your interface:

- [Amazon Redshift](#)
- [DB2 under UNIX and PC Hosts](#)
- [DB2 under z/OS](#)
- [Greenplum](#)
- [HAWQ](#)
- [Microsoft SQL Server](#)
- [ODBC](#)
- [OLE DB](#)
- [Oracle](#)
- [PostgreSQL](#)
- [SAP ASE](#)
- [SAP IQ](#)
- [Teradata](#)
- [Vertica](#)
- [Yellowbrick](#)

---

## UPDATE\_LOCK\_TYPE= LIBNAME Statement Option

Specifies how data in a DBMS table is locked during an update transaction.

|              |                                                                                                                                                                                                                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                                                                                                                                                                                                  |
| Category:    | Data Access                                                                                                                                                                                                                                                                                                   |
| Defaults:    | none [DB2 under z/OS, Teradata]<br>set by the data provider [OLE DB]<br>NOLOCK [Oracle]<br>PAGE [SAP ASE]<br>ROW [Amazon Redshift, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Microsoft SQL Server, ODBC, PostgreSQL, SAP ASE, SAP HANA IQ, Vertica, Yellowbrick]                                          |
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica, Yellowbrick                                                                                                              |
| Notes:       | Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                             |
| Tip:         | You can specify a lock for one DBMS table by using the data set option or for a group of DBMS tables by using the LIBNAME option.                                                                                                                                                                             |
| See:         | <a href="#">CONNECTION= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= data set option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= data set option</a> , <a href="#">UPDATE_LOCK_TYPE= data set</a> |

option, SCHEMA= LIBNAME option, and DBMS-specific locking information in the reference section for your SAS/ACCESS interface

## Syntax

**UPDATE\_LOCK\_TYPE=**[ROW](#) | [PAGE](#) | [TABLE](#) | [NOLOCK](#) | [VIEW](#)

### Syntax Description

#### **ROW**

locks a row if any of its columns are to be updated.

**Data source** Amazon Redshift, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Microsoft SQL Server, ODBC, Oracle, PostgreSQL, SAP HANA, Teradata, Vertica

#### **PAGE**

locks a page of data. The number of bytes in a page is specific to the DBMS.

**Data source** SAP ASE

#### **TABLE**

locks the entire DBMS table.

**Data source** DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, Oracle, SAP HANA, SAP IQ, Teradata

#### **NOLOCK**

does not lock the DBMS table, page, or any rows when reading them for update.

**Data source** Amazon Redshift, Microsoft SQL Server, ODBC with Microsoft SQL Server driver, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA

#### **VIEW**

locks the entire DBMS view.

**Data source** Teradata

## UPDATE\_MODE\_WAIT= LIBNAME Option LIBNAME Statement Option

Specifies during SAS/ACCESS Update operations whether Teradata should wait to acquire a lock or fail the request when a different user has locked the DBMS resource.

Valid in: SAS/ACCESS

Category: Data Access

Default: none

Data source: Teradata

See: [Locking in the Teradata Interface, UPDATE\\_MODE\\_WAIT= data set option](#)

## Syntax

**UPDATE\_MODE\_WAIT=YES | NO**

### Syntax Description

#### YES

specifies for Teradata to wait to acquire the lock, so SAS/ACCESS waits indefinitely until it can acquire the lock.

#### NO

specifies that Teradata fails the lock request if the specified DBMS resource is locked.

## Details

If you specify UPDATE\_MODE\_WAIT=NO and a different user holds a restrictive lock, the executing SAS step fails. SAS/ACCESS continues processing the job by executing the next step.

A *restrictive* lock means that a different user is holding a lock that prevents you from obtaining the lock that you want. Until the other user releases the restrictive lock, you cannot obtain your lock. For example, another user's table-level WRITE lock prevents you from obtaining a READ lock on the table.

Use SAS/ACCESS locking options only when the standard Teradata standard locking is undesirable.

## UPDATE\_MULT\_ROWS= LIBNAME Statement Option

Indicates whether SAS updates multiple rows from a data source, such as a DBMS table.

Valid in: SAS/ACCESS LIBNAME statement

Category: Data Set Control

Default: NO

Requirement: Snowflake: To use DELETE or UPDATE statements for multiple rows in a table, the table must have a primary key, and the data for the primary key must be unique. Otherwise, you must set the UPDATE\_MULT\_ROWS= LIBNAME option to YES.

Data source: Aster, Greenplum, HAWQ, Impala, Microsoft SQL Server, ODBC, OLE DB, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick

- Notes:      Support for HAWQ was added in SAS 9.4M3.  
               Support for Impala was added in SAS Viya 3.4.  
               Support for Snowflake was added in the August 2019 release of SAS/ACCESS.  
               Support for Yellowbrick was added in SAS 9.4M7.
- See:        [DELETE\\_MULT\\_ROWS= LIBNAME option](#)

## Syntax

**UPDATE\_MULT\_ROWS=YES | NO**

### Syntax Description

#### YES

specifies that SAS/ACCESS processing continues if multiple rows are updated.  
    This might produce unexpected results.

#### NO

specifies that SAS/ACCESS processing does not continue if multiple rows are updated.

## Details

Some providers do not handle this DBMS SQL statement well and therefore update more than the current row with this statement:

UPDATE...WHERE CURRENT OF CURSOR

UPDATE\_MULT\_ROWS= allows SAS/ACCESS to continue if multiple rows were updated.

## UPDATE\_SQL= LIBNAME Statement Option

Determines the method that is used to update and delete rows in a data source.

- Valid in:     SAS/ACCESS LIBNAME statement
- Category:    Data Set Control
- Defaults:    YES [except for the Oracle drivers from Microsoft and Oracle]  
               NO [Oracle drivers from Microsoft and Oracle, which do not support Current-of-Cursor operations]
- Data source:   Greenplum, HAWQ, Microsoft SQL Server, ODBC, Vertica
- Note:        Support for HAWQ was added in SAS 9.4M3.
- See:        [INSERT\\_SQL= LIBNAME option](#), [UPDATE\\_SQL= data set option](#)

## Syntax

**UPDATE\_SQL=YES | NO**

### Syntax Description

#### YES

specifies that SAS/ACCESS uses Current-of-Cursor SQL to update or delete rows in a table.

#### NO

specifies that SAS/ACCESS uses the SQLSetPos() application programming interface (API) to update or delete rows in a table.

---

## Details

This is the equivalent of update or delete for the [INSERT\\_SQL= LIBNAME](#) option. The default for Oracle drivers from Microsoft and Oracle is NO because these drivers do not support Current-Of-Cursor operations.

---

## UPDATEBUFF= LIBNAME Statement Option

Specifies the number of rows that are processed in a single DBMS Update or Delete operation.

|              |                                             |
|--------------|---------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                |
| Category:    | Data Set Control                            |
| Default:     | 1                                           |
| Data source: | Oracle                                      |
| See:         | <a href="#">UPDATEBUFF= data set option</a> |

---

## Syntax

**UPDATEBUFF=*positive-integer***

### Syntax Description

#### *positive-integer*

the number of rows in an operation. SAS allows the maximum that the DBMS allows.

## Details

When updating with the VIEWTABLE window or the FSVIEW procedure, use UPDATEBUFF=1 to prevent the DBMS interface from trying to update multiple rows. By default, these features update only one observation at a time. They do this because they use record-level locking by default and therefore lock only the observation that is currently being edited.

---

## USE\_DATADIRECT= LIBNAME Statement Option

Specifies the driver to use.

|              |                              |
|--------------|------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement |
| Category:    | Data Access                  |
| Alias:       | USE_PROGRESS=                |
| Default:     | NO                           |
| Data source: | Impala                       |

---

## Syntax

**USE\_DATADIRECT=YES | NO**

### Syntax Description

#### **YES**

specifies to use the DataDirect driver.

#### **NO**

specifies to use the Cloudera driver.

---

## USE\_INFORMATION\_SCHEMA= LIBNAME Statement Option

Specifies whether to use INFORMATION\_SCHEMA in Google BigQuery to retrieve a list of schemas and tables or views.

|              |                                                                             |
|--------------|-----------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                |
| Category:    | Data Access                                                                 |
| Default:     | YES                                                                         |
| Data source: | Google BigQuery                                                             |
| Note:        | Support for this option was added in the August 2019 release of SAS/ACCESS. |

## Syntax

**USE\_INFORMATION\_SCHEMA=YES | NO**

## Details

By default, SAS/ACCESS uses INFORMATION\_SCHEMA to get metadata for the list of schemas and tables or views, if the names contain a pattern. Names with patterns can be searched using the wildcard characters '%' or '\_'.

You might want to disable INFORMATION\_SCHEMA to improve performance, especially if you are not using patterns to query schema or table names.

---

**Note:** SAS directory services, including calls to PROC DATASETS, do not use patterns to retrieve schema or table names.

---

## USE\_ODBC\_CL= LIBNAME Statement Option

Indicates whether the Driver Manager uses the ODBC Cursor Library.

|              |                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement and some DBMS-specific connection options. See the DBMS-specific reference section for details. |
| Category:    | Data Access                                                                                                                  |
| Default:     | NO                                                                                                                           |
| Data source: | Aster, Microsoft SQL Server, ODBC, SAP HANA                                                                                  |
| See:         | For more information about the ODBC Cursor Library, see your vendor-specific documentation.                                  |

## Syntax

**USE\_ODBC\_CL=****YES | NO**

## Syntax Description

### YES

specifies that the Driver Manager uses the ODBC Cursor Library. The ODBC Cursor Library supports block scrollable cursors and positioned update and delete statements.

### NO

specifies that the Driver Manager uses the scrolling capabilities of the driver.

## UTILCONN\_TRANSIENT= LIBNAME Statement Option

Enables utility connections to maintain or drop, as needed.

Valid in: SAS/ACCESS LIBNAME statement and some DBMS-specific connection options. See the DBMS-specific reference section for details.

Category: Data Access

Defaults: YES [DB2 under z/OS]

NO [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick]

Restriction: UTILCONN\_TRANSIENT= has no effect on engines that do not support utility connections.

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

See: [DELETE\\_MULT\\_ROWS= LIBNAME option](#)

## Syntax

**UTILCONN\_TRANSIENT=NO | YES**

## Syntax Description

### NO

specifies that a utility connection is maintained for the lifetime of the libref.

### YES

specifies that a utility connection is automatically dropped as soon as it is no longer in use.

## Details

A utility connection is used for engines that can lock system resources as a result of such operations as DELETE or RENAME. Queries on system tables or table indexes can also lock system resources. This connection prevents COMMIT

statements that are issued to unlock system resources from being submitted on the same connection that is being used for table processing. Keeping COMMIT statements from table processing connection alleviates such problems that they can cause as invalidating cursors and committing pending updates on the tables that are being processed.

Because a utility connection exists for each LIBNAME statement, the number of connections to a DBMS can be large as multiple librefs are assigned across multiple SAS sessions. Specifying UTILCONN\_TRANSIENT=YES keeps these connections from existing when they are not being used. This value reduces the number of current connections to the DBMS at any given point in time.

---

## WARN\_BIGINT= LIBNAME Statement Option

issues a warning in the SAS log if the BIGINT data type is in a DBMS table.

|              |                                                             |
|--------------|-------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                |
| Category:    | Data Set Control                                            |
| Alias:       | BIGINT_WARN=                                                |
| Default:     | NO                                                          |
| Data source: | DB2 under UNIX and PC Hosts, Microsoft SQL Server           |
| Note:        | Support for Microsoft SQL Server was added in SAS Viya 3.4. |

---

## Syntax

**WARN\_BIGINT=YES | NO**

### Syntax Description

#### **YES**

issues a warning in the SAS log if a BIGINT data type is in a DBMS table.

#### **NO**

specifies that the warning for a BIGINT data type does not appear in the SAS log.

---

## Details

If a BIGINT data type is detected in the result set and if **WARN\_BIGINT=YES**, the log contains this warning:

A column of type BIGINT was detected in the result set. As BIGINT values are stored in SAS as DOUBLE PRECISION values, you may receive inexact results if the BIGINT value has a precision greater than 15 digits. Consider using the DBSASTYPE option to convert the BIGINT column into character value to preserve the precision of the BIGINT value.



# Data Set Options

---

|                                                                  |            |
|------------------------------------------------------------------|------------|
| <b>About the Data Set Options for Relational Databases .....</b> | <b>370</b> |
| Overview .....                                                   | 370        |
| <b>Dictionary .....</b>                                          | <b>371</b> |
| ANALYZE= Data Set Option .....                                   | 371        |
| AUTHID= Data Set Option .....                                    | 372        |
| AUTOCOMMIT= Data Set Option .....                                | 373        |
| BL_ACCOUNTNAME= Data Set Option .....                            | 373        |
| BL_ALLOW_READ_ACCESS= Data Set Option .....                      | 374        |
| BL_ALLOW_WRITE_ACCESS= Data Set Option .....                     | 375        |
| BL_API_BULKLOAD= Data Set Option .....                           | 376        |
| BL_APPLICATIONID= Data Set Option .....                          | 377        |
| BL_AWS_CONFIG_FILE= Data Set Option .....                        | 378        |
| BL_AWS_CREDENTIALS_FILE= Data Set Option .....                   | 378        |
| BL_AWS_PROFILE_NAME= Data Set Option .....                       | 379        |
| BL_BADFILE= Data Set Option .....                                | 380        |
| BL_BUCKET= Data Set Option .....                                 | 381        |
| BL_CLIENT_DATAFILE= Data Set Option .....                        | 382        |
| BL_CODEPAGE= Data Set Option .....                               | 382        |
| BL_COLUMN_DELIMITER= Data Set Option .....                       | 383        |
| BL_COMPRESS= Data Set Option .....                               | 384        |
| BL_CONFIG= Data Set Option .....                                 | 384        |
| BL_CONTROL_FIELD_DELIMITER= Data Set Option .....                | 386        |
| BL_CONTROL_QUOTATION_MARK= Data Set Option .....                 | 386        |
| BL_CONTROL_RECORD_DELIMITER= Data Set Option .....               | 387        |
| BL_CONTROL= Data Set Option .....                                | 388        |
| BL_COPY_LOCATION= Data Set Option .....                          | 390        |
| BL_CPU_PARALLELISM= Data Set Option .....                        | 390        |
| BL_DATA_BUFFER_SIZE= Data Set Option .....                       | 392        |
| BL_DATAFILE_EXISTS= Data Set Option .....                        | 393        |
| BL_DATAFILE_PATH= Data Set Option .....                          | 393        |
| BL_DATAFILE= Data Set Option .....                               | 394        |
| BL_DATAFILE= Data Set Option: Teradata .....                     | 396        |
| BL_DB2CURSOR= Data Set Option .....                              | 398        |
| BL_DB2DATACLAS= Data Set Option .....                            | 399        |
| BL_DB2DEVT_PERM= Data Set Option .....                           | 400        |
| BL_DB2DEVT_TEMP= Data Set Option .....                           | 400        |
| BL_DB2DISC= Data Set Option .....                                | 401        |

|                                                |     |
|------------------------------------------------|-----|
| BL_DB2ERR= Data Set Option .....               | 401 |
| BL_DB2IN= Data Set Option .....                | 402 |
| BL_DB2LDCT1= Data Set Option .....             | 403 |
| BL_DB2LDCT2= Data Set Option .....             | 404 |
| BL_DB2LDCT3= Data Set Option .....             | 404 |
| BL_DB2LDEXT= Data Set Option .....             | 405 |
| BL_DB2MGMTCLAS= Data Set Option .....          | 406 |
| BL_DB2MAP= Data Set Option .....               | 406 |
| BL_DB2PRINT= Data Set Option .....             | 407 |
| BL_DB2PRNLOG= Data Set Option .....            | 408 |
| BL_DB2REC= Data Set Option .....               | 408 |
| BL_DB2RECSPI= Data Set Option .....            | 409 |
| BL_DB2RSTRT= Data Set Option .....             | 410 |
| BL_DB2SPC_PERM= Data Set Option .....          | 410 |
| BL_DB2SPC_TEMP= Data Set Option .....          | 411 |
| BL_DB2STORCLAS= Data Set Option .....          | 411 |
| BL_DB2TBLXST= Data Set Option .....            | 413 |
| BL_DB2UNITCOUNT= Data Set Option .....         | 414 |
| BL_DB2UTID= Data Set Option .....              | 415 |
| BL_DBNAME= Data Set Option .....               | 415 |
| BL_DEFAULT_DIR= Data Set Option .....          | 416 |
| BL_DELETE_DATAFILE= Data Set Option .....      | 417 |
| BL_DELETE_ONLY_DATAFILE= Data Set Option ..... | 419 |
| BL_DELIMITER= Data Set Option .....            | 421 |
| BL_DIRECT_PATH= Data Set Option .....          | 423 |
| BL_DISCARDFILE= Data Set Option .....          | 424 |
| BL_DISK_PARALLELISM= Data Set Option .....     | 425 |
| BL_DNSSUFFIX= Data Set Option .....            | 426 |
| BL_ENCKEY= Data Set Option .....               | 427 |
| BL_ENCODING= Data Set Option .....             | 428 |
| BL_ESCAPE= Data Set Option .....               | 429 |
| BL_EXCEPTION= Data Set Option .....            | 430 |
| BL_EXECUTE_CMD= Data Set Option .....          | 431 |
| BL_EXECUTE_LOCATION= Data Set Option .....     | 432 |
| BL_EXTERNAL_WEB= Data Set Option .....         | 433 |
| BL_FILE_BADFILE= Data Set Option .....         | 434 |
| BL_FILE_CONTROLFILE= Data Set Option .....     | 435 |
| BL_FILE_DATAFILE= Data Set Option .....        | 436 |
| BL_FILE_DEFAULT_DIR= Data Set Option .....     | 437 |
| BL_FILE_DELETE_DATAFILE= Data Set Option ..... | 437 |
| BL_FILESYSTEM= Data Set Option .....           | 438 |
| BL_FOLDER= Data Set Option .....               | 439 |
| BL_FORCE_NOT_NULL= Data Set Option .....       | 440 |
| BL_FORMAT= Data Set Option .....               | 441 |
| BL_FORMAT_OPT= Data Set Option .....           | 442 |
| BL_HEADER= Data Set Option .....               | 442 |
| BL_HOST= Data Set Option .....                 | 443 |
| BL_IAM_ROLE= Data Set Option .....             | 444 |
| BL_IDENTITY= Data Set Option .....             | 445 |
| BL_IMPORT_BATCH_SIZE= Data Set Option .....    | 446 |
| BL_IMPORT_OPTIONS= Data Set Option .....       | 446 |
| BL_IMPORT_TABLE_LOCK= Data Set Option .....    | 447 |
| BL_IMPORT_TYPE_CHECK= Data Set Option .....    | 448 |
| BL_INDEX_OPTIONS= Data Set Option .....        | 448 |

|                                                     |     |
|-----------------------------------------------------|-----|
| BL_INDEXING_MODE= Data Set Option .....             | 450 |
| BL_INTERNAL_STAGE= Data Set Option .....            | 450 |
| BL_KEEPIDENTITY= Data Set Option .....              | 451 |
| BL_KEEPNULLS= Data Set Option .....                 | 452 |
| BL_KEY= Data Set Option .....                       | 452 |
| BL_LOAD_METHOD= Data Set Option .....               | 453 |
| BL_LOAD_REPLACE= Data Set Option .....              | 454 |
| BL_LOCATION= Data Set Option .....                  | 454 |
| BL_LOG= Data Set Option .....                       | 455 |
| BL_LOGFILE= Data Set Option .....                   | 456 |
| BL_MAPFILE= Data Set Option .....                   | 457 |
| BL_MAXERRORS= Data Set Option .....                 | 458 |
| BL_METHOD= Data Set Option .....                    | 459 |
| BL_NULL= Data Set Option .....                      | 459 |
| BL_NUM_DATAFILES= Data Set Option .....             | 460 |
| BL_NUM_READ_THREADS= Data Set Option .....          | 461 |
| BL_OPTIONS= Data Set Option .....                   | 462 |
| BL_PARFILE= Data Set Option .....                   | 464 |
| BL_PATH= Data Set Option .....                      | 465 |
| BL_PORT= Data Set Option .....                      | 466 |
| BL_PORT_MAX= Data Set Option .....                  | 467 |
| BL_PORT_MIN= Data Set Option .....                  | 468 |
| BL_PRESERVE_BLANKS= Data Set Option .....           | 468 |
| BL_PROTOCOL= Data Set Option .....                  | 469 |
| BL_PSQL_PATH= Data Set Option .....                 | 470 |
| BL_QUOTE= Data Set Option .....                     | 470 |
| BL_RECOVERABLE= Data Set Option .....               | 471 |
| BL_REGION= Data Set Option .....                    | 472 |
| BL_REJECT_LIMIT= Data Set Option .....              | 473 |
| BL_REJECT_TYPE= Data Set Option .....               | 474 |
| BL_REMOTE_FILE= Data Set Option .....               | 475 |
| BL_RETURN_WARNINGS_AS_ERRORS= Data Set Option ..... | 476 |
| BL_ROW_DELIMITER= Data Set Option .....             | 476 |
| BL_SECRET= Data Set Option .....                    | 477 |
| BL_SERVER_DATAFILE= Data Set Option .....           | 478 |
| BL_SFTP_HOST= Data Set Option .....                 | 479 |
| BL_SFTP_OPTIONS= Data Set Option .....              | 480 |
| BL_SFTP_USER= Data Set Option .....                 | 481 |
| BL_SFTP_WAIT_MILLISECONDS= Data Set Option .....    | 481 |
| BL_SUPPRESS_NULLIF= Data Set Option .....           | 482 |
| BL_TIMEOUT= Data Set Option .....                   | 483 |
| BL_TOKEN= Data Set Option .....                     | 484 |
| BL_USE_MANIFEST= Data Set Option .....              | 485 |
| BL_USE_ESCAPE= Data Set Option .....                | 485 |
| BL_USE_LOG= Data Set Option .....                   | 486 |
| BL_USE_PIPE= Data Set Option .....                  | 487 |
| BL_USE_SSL= Data Set Option .....                   | 488 |
| BL_WARNING_COUNT= Data Set Option .....             | 489 |
| BL_YB_PATH= Data Set Option .....                   | 489 |
| BUFFERS= Data Set Option Data Set Option .....      | 490 |
| BULK_BUFFER= Data Set Option .....                  | 491 |
| BULKLOAD= Data Set Option .....                     | 491 |
| BULKUNLOAD= Data Set Option .....                   | 493 |
| BUSINESS_DATATYPE= Data Set Option .....            | 494 |

|                                                  |     |
|--------------------------------------------------|-----|
| BUSINESS_TIMEFRAME= Data Set Option .....        | 495 |
| CAST= Data Set Option .....                      | 495 |
| CAST_OVERHEAD_MAXPERCENT= Data Set Option .....  | 497 |
| CHAR_AS_BINARY= Data Set Option .....            | 498 |
| COLUMN_DELIMITER= Data Set Option .....          | 498 |
| COMMAND_TIMEOUT= Data Set Option .....           | 499 |
| CURSOR_TYPE= Data Set Option .....               | 499 |
| DATETIME2= Data Set Option .....                 | 502 |
| DB_ONE_CONNECT_PER_THREAD= Data Set Option ..... | 502 |
| DBCOMMIT= Data Set Option .....                  | 503 |
| DBCONDITION= Data Set Option .....               | 504 |
| DBCONSTRAINT= Data Set Option .....              | 506 |
| DBCREATE_TABLE_EXTERNAL= Data Set Option .....   | 507 |
| DBCREATE_TABLE_LOCATION= Data Set Option .....   | 508 |
| DBCREATE_TABLE_OPTS= Data Set Option .....       | 509 |
| DBFORCE= Data Set Option .....                   | 511 |
| DBGEN_NAME= Data Set Option .....                | 513 |
| DBINDEX= Data Set Option .....                   | 514 |
| DBKEY= Data Set Option .....                     | 516 |
| DBLABEL= Data Set Option .....                   | 518 |
| DBLINK= Data Set Option .....                    | 519 |
| DBLARGETABLE= Data Set Option .....              | 520 |
| DBMAX_TEXT= Data Set Option .....                | 521 |
| DBNULL= Data Set Option .....                    | 522 |
| DBNULLKEYS= Data Set Option .....                | 524 |
| DBNULLWHERE= Data Set Option .....               | 525 |
| DBPROMPT= Data Set Option .....                  | 526 |
| DBSASLABEL= Data Set Option .....                | 528 |
| DBSASTYPE= Data Set Option .....                 | 529 |
| DBSLICE= Data Set Option .....                   | 531 |
| DBSLICEPARM= Data Set Option .....               | 533 |
| DBTYPE= Data Set Option .....                    | 536 |
| DEGREE= Data Set Option Data Set Option .....    | 539 |
| DIMENSION= Data Set Option .....                 | 540 |
| DISTRIBUTE_ON= Data Set Option .....             | 540 |
| DISTRIBUTED_BY= Data Set Option .....            | 542 |
| ERRLIMIT= Data Set Option .....                  | 543 |
| ESCAPE_BACKSLASH= Data Set Option .....          | 544 |
| FASTEXPORT= Data Set Option .....                | 546 |
| FASTLOAD= Data Set Option .....                  | 547 |
| FETCH_IDENTITY= Data Set Option .....            | 549 |
| HDFS_PRINCIPAL= Data Set Option .....            | 550 |
| IGNORE_READ_ONLY_COLUMNS= Data Set Option .....  | 551 |
| IN= Data Set Option .....                        | 552 |
| INSERT_SQL= Data Set Option .....                | 553 |
| INSERTBUFF= Data Set Option .....                | 554 |
| KEYSET_SIZE= Data Set Option .....               | 555 |
| LOCATION= Data Set Option .....                  | 556 |
| LOCKTABLE= Data Set Option .....                 | 557 |
| MBUFSIZE= Data Set Option .....                  | 557 |
| ML_CHECKPOINT= Data Set Option .....             | 558 |
| ML_ERROR1= Data Set Option .....                 | 559 |
| ML_ERROR2= Data Set Option .....                 | 560 |
| ML_LOG= Data Set Option .....                    | 561 |

|                                                     |     |
|-----------------------------------------------------|-----|
| ML_RESTART= Data Set Option .....                   | 562 |
| ML_WORK= Data Set Option .....                      | 563 |
| MULTILOAD= Data Set Option .....                    | 564 |
| MULTISTMT= Data Set Option .....                    | 567 |
| NULCHAR= Data Set Option .....                      | 569 |
| NULCHARVAL= Data Set Option .....                   | 571 |
| OR_IDENTITY_COLS= Data Set Option .....             | 571 |
| OR_PARTITION= Data Set Option .....                 | 572 |
| OR_UPD_NOWHERE= Data Set Option .....               | 576 |
| ORHINTS= Data Set Option .....                      | 577 |
| OVERLAPS= Data Set Option .....                     | 578 |
| PARMDEFAULT= Data Set Option .....                  | 579 |
| PARMSTRING= Data Set Option .....                   | 580 |
| PARTITION_KEY= Data Set Option .....                | 581 |
| PARTITIONED_BY= Data Set Option .....               | 582 |
| PG_IDENTITY_COLS= Data Set Option .....             | 583 |
| POST_DML_STMT_OPTS= Data Set Option .....           | 584 |
| POST_STMT_OPTS= Data Set Option .....               | 585 |
| POST_TABLE_OPTS= Data Set Option .....              | 587 |
| PRE_STMT_OPTS= Data Set Option .....                | 588 |
| PRE_TABLE_OPTS= Data Set Option .....               | 590 |
| PRESERVE_COL_NAMES= Data Set Option .....           | 591 |
| QUALIFIER= Data Set Option .....                    | 593 |
| QUERY_BAND= Data Set Option .....                   | 594 |
| QUERY_TIMEOUT= Data Set Option .....                | 595 |
| READBUFF= Data Set Option .....                     | 595 |
| READ_ISOLATION_LEVEL= Data Set Option .....         | 597 |
| READ_LOCK_TYPE= Data Set Option .....               | 597 |
| READ_METHOD= Data Set Option .....                  | 599 |
| READ_MODE= Data Set Option .....                    | 600 |
| READ_MODE_WAIT= Data Set Option .....               | 601 |
| ROW_DELIMITER= Data Set Option .....                | 601 |
| SASDATEFMT= Data Set Option .....                   | 602 |
| SCANSTRINGCOLUMNS= Data Set Option .....            | 604 |
| SCHEMA= Data Set Option .....                       | 605 |
| SCRATCH_DB= Data Set Option .....                   | 607 |
| SEGMENT_NAME= Data Set Option Data Set Option ..... | 608 |
| SESSIONS= Data Set Option .....                     | 608 |
| SET= Data Set Option .....                          | 609 |
| SLEEP= Data Set Option .....                        | 610 |
| SUB_CHAR= Data Set Option .....                     | 611 |
| SYSTEM_TIMEFRAME= Data Set Option .....             | 612 |
| TABLE_TYPE= Data Set Option .....                   | 613 |
| TEMPORAL= Data Set Option .....                     | 614 |
| TEMPORAL_QUALIFIER= Data Set Option .....           | 615 |
| TENACITY= Data Set Option .....                     | 618 |
| TPT= Data Set Option .....                          | 619 |
| TPT_APPL_PHASE= Data Set Option .....               | 620 |
| TPT_BLOCK_SIZE= Data Set Option .....               | 622 |
| TPT_BUFFER_SIZE= Data Set Option .....              | 623 |
| TPT_CHECKPOINT_DATA= Data Set Option .....          | 623 |
| TPT_DATA_ENCRYPTION= Data Set Option .....          | 626 |
| TPT_ERROR_TABLE_1= Data Set Option .....            | 627 |
| TPT_ERROR_TABLE_2= Data Set Option .....            | 628 |

|                                               |     |
|-----------------------------------------------|-----|
| TPT_LOG_TABLE= Data Set Option .....          | 629 |
| TPT_MAX_SESSIONS= Data Set Option .....       | 631 |
| TPT_MIN_SESSIONS= Data Set Option .....       | 632 |
| TPT_PACK= Data Set Option .....               | 632 |
| TPT_PACKMAXIMUM= Data Set Option .....        | 633 |
| TPT_RESTART= Data Set Option .....            | 634 |
| TPT_TRACE_LEVEL= Data Set Option .....        | 637 |
| TPT_TRACE_LEVEL_INF= Data Set Option .....    | 638 |
| TPT_TRACE_OUTPUT= Data Set Option .....       | 639 |
| TPT_WORK_TABLE= Data Set Option .....         | 640 |
| TRANSCODE_FAIL= Data Set Option .....         | 641 |
| TRAP151= Data Set Option .....                | 642 |
| UPDATE_ISOLATION_LEVEL= Data Set Option ..... | 644 |
| UPDATE_LOCK_TYPE= Data Set Option .....       | 644 |
| UPDATE_MODE_WAIT= Data Set Option .....       | 646 |
| UPDATE_SQL= Data Set Option .....             | 647 |
| UPDATEBUFF= Data Set Option .....             | 647 |
| UPSERT= Data Set Option .....                 | 648 |
| UPSERT_CONDITION= Data Set Option .....       | 650 |
| UPSERT_WHERE= Data Set Option .....           | 651 |
| YB_JAVA_HOME= Data Set Option .....           | 653 |

---

## About the Data Set Options for Relational Databases

---

### Overview

You can specify SAS/ACCESS data set options on a SAS data set when you access DBMS data with the SAS/ACCESS [LIBNAME statement](#). A data set option applies only to the data set on which it is specified, and it remains in effect for the duration of the DATA step or procedure. For options that you can assign to a group of relational DBMS tables or views, see “[LIBNAME Options for Relational Databases](#)” on page 134.

Here is an example of how you can specify SAS/ACCESS data set options.

```
libname mydblib hadoop;
proc print mydblib.mytable(data-set-option=value);
```

You can also use SAS/ACCESS data set options on a SAS data set when you access DBMS data using access descriptors. See “[Overview](#)” on page 370. Here is an example.

```
proc print mylib.myviewd(data-set-option=value)
```

You cannot use most data set options in a PROC SQL DROP (table or view) statement.

You can use the CNTLLEV=, DROP=, FIRSTOBS=, IN=, KEEP=, OBS=, RENAME=, and WHERE= SAS data set options when you access DBMS data.

SAS/ACCESS interfaces do not support the REPLACE= SAS data set option. For information about using SAS data set options, see *SAS Data Set Options: Reference*.

The information in this section explains all applicable data set options. The information includes DBMS support and the corresponding LIBNAME options, and it refers you to documentation for your SAS/ACCESS interface when appropriate. For a list of the data set options available in your SAS/ACCESS interface with default values, see the DBMS-specific reference section for your SAS/ACCESS interface.

Specifying data set options in PROC SQL might reduce performance because it prevents operations from being passed to the DBMS for processing. For more information, see “[Overview: Optimizing Your SQL Usage](#)” on page 55.

---

## Dictionary

---

### ANALYZE= Data Set Option

Lets SAS improve performance when a single Hive store table is queried.

|              |                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                             |
| Category:    | Data Set Control                                                                                                                     |
| Default:     | NO                                                                                                                                   |
| Data source: | Hadoop                                                                                                                               |
| See:         | <a href="#">ANALYZE= LIBNAME option</a> , <a href="#">READ_METHOD= LIBNAME option</a> , <a href="#">READ_METHOD= data set option</a> |

---

## Syntax

**ANALYZE=YES | NO**

### Syntax Description

#### **YES**

specifies that SAS might run an ANALYZE TABLE command to update table statistics. Current table statistics might improve SAS Read performance when a single table is queried. This operation is considered a hint, so if statistics are up-to-date, SAS might not perform the operation. The format of the ANALYZE TABLE command is subject to change in future releases of SAS as needed.

#### **NO**

specifies that SAS does not perform an additional operation to update table statistics.

## Details

Performance improves when Hive statistics are up-to-date. When the Hive Boolean variable `hive.stats.autogather` is set to TRUE, in most cases Hive automatically gathers statistics. This option can be useful when `hive.stats.autogather` is set to FALSE or when statistics are not being computed. Specifically, Hive statistics are not generated when loading a target table with a file format of TEXT. Users can check the state of Hive statistics using the Hive DESCRIBE FORMATTED command.

---

## AUTHID= Data Set Option

Lets you qualify the specified table with an authorization ID, user ID, or group ID.

|              |                                                                                    |
|--------------|------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)           |
| Category:    | Data Set Control                                                                   |
| Alias:       | SCHEMA=                                                                            |
| Default:     | LIBNAME option value                                                               |
| Data source: | DB2 under z/OS                                                                     |
| See:         | <a href="#">AUTHID= LIBNAME option</a> , <a href="#">DBCONINIT= LIBNAME option</a> |

---

## Syntax

**AUTHID=*authorization-ID***

### Syntax Description

***authorization-ID***

limited to eight characters.

---

## Details

If you specify a value for the AUTHID= option, the table name is qualified as `authid.tablename` before any SQL code is passed to the DBMS. If you do not specify AUTHID=, the table name is not qualified before being passed to the DBMS and the DBMS uses your user ID as the qualifier. If you specify AUTHID= in a SAS/SHARE LIBNAME statement, the ID of the active server is the default ID.

If you specify DBCONINIT="SET CURRENT SQLID='*user-ID*'", then any value that is specified for AUTHID= or SCHEMA= is ignored.

---

## AUTOCOMMIT= Data Set Option

Specifies whether to enable the DBMS autocommit capability.

|              |                                                                                        |
|--------------|----------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)               |
| Category:    | Data Set Control                                                                       |
| Default:     | LIBNAME option value                                                                   |
| Data source: | SAP ASE                                                                                |
| See:         | <a href="#">AUTOCOMMIT= LIBNAME option</a> , <a href="#">DBCOMMIT= data set option</a> |

---

## Syntax

**AUTOCOMMIT=YES | NO**

### Syntax Description

#### **YES**

specifies that SAS commits all updates, inserts, and deletes immediately after they are executed and that no rollback is possible.

#### **NO**

specifies that SAS commits after processing the number of rows that DBCOMMIT= specifies. If you do not specify DBCOMMIT=, SAS commits after processing the default number of rows.

---

## BL\_ACCOUNTNAME= Data Set Option

Specifies the name of the account to use on the external storage system.

|              |                                                                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                        |
| Categories:  | Data Access<br>Bulk Loading                                                                                                                                                                     |
| Default:     | LIBNAME option value                                                                                                                                                                            |
| Interaction: | This option is used with the BL_DNSSUFFIX=, BL_FILESYSTEM=, and BL_FOLDER= options to specify the external data location.                                                                       |
| Data source: | Microsoft SQL Server                                                                                                                                                                            |
| Note:        | Support for this data set option was added in SAS 9.4M7.                                                                                                                                        |
| See:         | <a href="#">BL_ACCOUNTNAME= LIBNAME option</a> , <a href="#">BL_DNSSUFFIX= data set option</a> ,<br><a href="#">BL_FILESYSTEM= data set option</a> , <a href="#">BL_FOLDER= data set option</a> |

## Syntax

**BL\_ACCOUNTNAME="account-name"**

### Required Argument

***account-name***

specifies the name of the account to use on the external storage system.

### Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

The external data location is generated using this option and the BL\_DNSSUFFIX=, BL\_FILESYSTEM=, and BL\_FOLDER= data set options. These values are combined to define the URL to access an Azure Data Lake Storage Gen2 location :

`https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>`

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

---

## BL\_ALLOW\_READ\_ACCESS= Data Set Option

Specifies that the original table data is still visible to readers during bulk loading.

|              |                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                           |
| Categories:  | Bulk Loading<br>Data Set Control                                                                   |
| Default:     | NO                                                                                                 |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                       |
| Data source: | DB2 under UNIX and PC Hosts                                                                        |
| See:         | <a href="#">BL_ALLOW_WRITE_ACCESS= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_ALLOW\_READ\_ACCESS=YES | NO**

## Syntax Description

### **YES**

specifies that the original (unchanged) data in the table is still visible to readers while bulk loading is in progress.

### **NO**

specifies that readers cannot view the original data in the table while bulk loading is in progress.

## Details

For more information about using this option, see the SQLU\_ALLOW\_READ\_ACCESS parameter in the *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*.

## BL\_ALLOW\_WRITE\_ACCESS= Data Set Option

Specifies that table data is still accessible to readers and writers while import is in progress.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: NO

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under UNIX and PC Hosts

See: [BL\\_ALLOW\\_READ\\_ACCESS= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_ALLOW\_WRITE\_ACCESS=**[YES | NO](#)

## Syntax Description

### **YES**

specifies that table data is still visible to readers and writers during data import.

### **NO**

specifies that readers and writers cannot view table data during data import.

## Details

For more information about using this option, see the SQLU\_ALLOW\_WRITE\_ACCESS parameter in the *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*.

---

## BL\_API\_BULKLOAD= Data Set Option

Specifies to perform bulk loading using the Oracle Direct Path API instead of the Oracle SQL\*Loader utility.

Valid in: DATA and PROC steps (when accessing data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: NO

Restriction: This data set option is not valid in z/OS operating environments.

Requirement: To specify this option, you must first set BULKLOAD=YES

Data source: Oracle

Note: Support for this data set option was added in SAS 9.4M2.

See: [BULKLOAD= data set option](#), [ERRLIMIT= data set option](#), [INSERTBUFF= data set option](#)

---

## Syntax

**BL\_API\_BULKLOAD=YES | NO**

### Syntax Description

#### YES

specifies to perform bulk loading using the Direct Path API.

**Notes** The Oracle SQL\* Loader is not required for bulk loading using Direct Path API.

Direct Path API bulk loading shows better performance for SAS tables that contain mainly numeric values as compared to the same bulk loading using the SQL\*Loader utility.

#### NO

specifies to perform bulk loading using the Oracle SQL\*Loader utility.

---

## Details

The following are differences in bulk loading when you use the Direct Path API:

- DAT files are not needed. Therefore, they are not created.
- The SLQ\*Loader utility is not used.
- Other bulk-loading data set options are ignored for Direct Path API bulk loading.
- Direct Path API bulk loading uses the value of the INSERTBUFF= data set option to optimize performance. It uses the ERRLIMIT= data set option to limit the number of errors before SAS stops processing and issues a rollback.

---

## BL\_APPLICATIONID= Data Set Option

Specifies the application ID (a GUID string) for accessing the external storage system.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Access

Default: LIBNAME option value

Data source: Microsoft SQL Server

Note: Support for this data set option was added in SAS 9.4M7.

See: [BL\\_APPLICATIONID= LIBNAME option](#)

Example: bl\_applicationid="b1fc955d5c-e0e2-45b3-a3cc-a1cf54120f"

---

## Syntax

**BL\_APPLICATIONID="value"**

### Required Argument

**value**

specifies the application ID (a GUID string) for accessing the external storage system.

---

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with a Microsoft Azure Data Lake Storage Gen2 storage account. For more information, see [“Bulk Loading to Azure Synapse Analytics” on page 1025](#).

---

## BL\_AWS\_CONFIG\_FILE= Data Set Option

Specifies the location of the AWS configuration file.

|              |                                                                                                                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                        |
| Category:    | Bulk Loading                                                                                                                                                                                                    |
| Alias:       | BL_AWS_CONFIG=                                                                                                                                                                                                  |
| Default:     | ~/.aws/config [Amazon Redshift, Snowflake], \\.aws\\config [Snowflake]                                                                                                                                          |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                    |
| Data source: | Amazon Redshift, Snowflake                                                                                                                                                                                      |
| Notes:       | Support for this data set option was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.                                                                        |
| See:         | <a href="#">BL_AWS_CONFIG_FILE= LIBNAME option</a> , <a href="#">BL_AWS_CONFIG_FILE= data set option</a> , <a href="#">BL_AWS_CREDENTIALS_FILE= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_AWS\_CONFIG\_FILE=***path-and-file-name*

### Required Argument

***path-and-file-name***

specifies the location of the AWS configuration file. By default, this location is  
~/.aws/config.

---

## BL\_AWS\_CREDENTIALS\_FILE= Data Set Option

Specifies the path to the credentials file.

|              |                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                             |
| Category:    | Bulk Loading                                                                                                                                         |
| Alias:       | BL_AWS_CRED_FILE=,BL_AWS_CREDENTIALS=                                                                                                                |
| Default:     | ~/.aws/credentials                                                                                                                                   |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                         |
| Data source: | Amazon Redshift, Snowflake                                                                                                                           |
| Note:        | Support for this data set option was added in the August 2019 release of SAS/ACCESS.                                                                 |
| See:         | <a href="#">BL_AWS_CONFIG_FILE= LIBNAME option</a> , <a href="#">BL_AWS_CONFIG_FILE= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_AWS\_CREDENTIALS\_FILE=***path-and-file-name*

### Required Argument

***path-and-file-name***

specifies the location of the AWS credentials file.

## BL\_AWS\_PROFILE\_NAME= Data Set Option

Specifies the AWS profile to use when there is more than one profile in the AWS configuration file.

|              |                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                 |
| Category:    | Bulk Loading                                                                                                                             |
| Alias:       | BL_AWS_PROFILE=                                                                                                                          |
| Default:     | none                                                                                                                                     |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                             |
| Data source: | Amazon Redshift, Snowflake                                                                                                               |
| Notes:       | Support for this data set option was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS. |
| See:         | <a href="#">BL_AWS_PROFILE_NAME= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a>                                          |

## Syntax

**BL\_AWS\_PROFILE\_NAME=***profile-name*

### Required Argument

***profile-name***

specifies the profile to use if there is more than one profile in the AWS configuration file.

## Details

If you specify more than one profile in your AWS configuration file, then each profile has a name that is specified in square brackets. Here is a sample configuration file with two profiles, default and analyst.

```
[default]
region=us-west-2
output=text
```

```
[analyst]
region=us-east-1
output=json
```

To use the analyst profile, specify the following code.

```
data myclass (bulkload=yes
              bl_bucket='myBucket/'
              bl_aws_profile=analyst
            );
set sashelp.class;
run;
```

## BL\_BADFILE= Data Set Option

Identifies a file that contains records that were rejected during bulk loading.

|              |                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                      |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                              |
| Default:     | creates a data file in the temporary file directory that is specified by the UTILLOC= system option or in another directory that is based on your default file specifications |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                  |
| Data source: | Oracle                                                                                                                                                                        |
| See:         | <a href="#">BULKLOAD= data set option</a>                                                                                                                                     |

## Syntax

**BL\_BADFILE=***path-and-file-name*

### Syntax Description

#### *path-and-file-name*

an SQL\*Loader file to which rejected rows of data are written. On most platforms, the default file name takes the form BL\_<table>\_<unique-ID>.bad:

#### *table*

specifies the table name.

#### *unique-ID*

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates the number.

---

## Details

If you do not specify this option and a BAD file does not exist, a file is created in the temporary file directory that is specified by the UTILLOC= system option (or in another directory that is based on your default file specifications). If you do not specify this option and a BAD file already exists, the Oracle bulk loader reuses the file, replacing the contents with rejected rows from the new load.

Either Oracle or the SQL\*Loader can reject records. For example, the SQL\*Loader can reject a record that contains invalid input, and Oracle can reject a record because it does not contain a unique key. If no records are rejected, the BAD file is not created.

On most operating systems, the BAD file is created in the same format as the DATA file, so the rejected records can be loaded after corrections have been made.

**Operating Environment Information:** On z/OS operating systems, the BAD file is created with default DCB attributes. For details about overriding this, see the information about SQL\*Loader file attributes in the SQL\*Loader chapter in your Oracle user's guide for z/OS.

---

## BL\_BUCKET= Data Set Option

Specifies the bucket name to use when bulk loading data files.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

Requirements: To specify this option, you must first specify BULKLOAD=YES.

The bucket name that you specify must already exist.

Interaction: When BULKLOAD=YES, the BL\_BUCKET= data set option is required. [Amazon Redshift]

Data source: Amazon Redshift, Google BigQuery, Snowflake

Notes: Support for this data set option was added in SAS 9.4M4.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Google BigQuery was added in SAS 9.4M7.

Tip: You can also use this option with the BULKUNLOAD= option.

See: [BL\\_BUCKET= LIBNAME option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_BUCKET='bucket-name'**

## Details

*Amazon Redshift, Snowflake:* This option refers to an Amazon S3 bucket. For more information, see [Amazon Web Services S3 information](#).

*Google BigQuery:* This option refers to a Google Cloud Storage bucket.

---

## BL\_CLIENT\_DATAFILE= Data Set Option

Specifies the client view of the data file that contains DBMS data for bulk loading.

|              |                                                                              |
|--------------|------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)     |
| Categories:  | Bulk Loading<br>Data Set Control                                             |
| Default:     | the temporary file directory that is specified by the UTILLOC= system option |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                 |
| Data source: | SAP IQ                                                                       |
| See:         | <a href="#">BULKLOAD= data set option</a>                                    |

---

## Syntax

**BL\_CLIENT\_DATAFILE=***path-and-data-file-name*

### Syntax Description

***path-and-data-file-name***

specifies the file that contains the rows of data to load or append into a DBMS table during bulk loading. On most platforms, the default file name takes the form *BL\_<table>\_<unique-ID>.dat*.

***table***

specifies the table name.

***unique-ID***

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk-loading operations on a particular table. The SAS/ACCESS engine generates the number.

***dat***

specifies the .dat file extension for the data file.

---

## BL\_CODEPAGE= Data Set Option

Identifies the code page that the DBMS engine uses to convert SAS character data to the current database code page during bulk loading.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: the code page ID of the current window

Requirements: The value for this option must never be 0. If you do not wish any code page conversions to take place, use the BL\_OPTIONS= option to specify 'FORCEIN'. Code page conversions occur for only DB2 character data types.  
To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under UNIX and PC Hosts

See: [BL\\_OPTIONS= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_CODEPAGE=***numeric-code-page-ID*

### Syntax Description

#### *numeric-code-page-ID*

a numeric value that represents a character set that is used to interpret multibyte character data and determine the character values.

## BL\_COLUMN\_DELIMITER= Data Set Option

Specifies override of the default delimiter character for separating columns of data during data transfer or retrieval during bulk loading.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Alias: BL\_DELIMITER=

Default: | (pipe symbol)

Data source: SAP IQ

Note: Support for this data set option was added in SAS 9.4M5.

See: [BL\\_DELIMITER= data set option](#), [BL\\_ROW\\_DELIMITER= data set option](#)

## Syntax

**BL\_COLUMN\_DELIMITER='***any-single-character***'**

## Required Argument

### ***any-single-character***

specifies a single character that is used to delimit columns in your data. You can specify the delimiter as a single printable character, or you can use hexadecimal notation to specify any single 8-bit hexadecimal ASCII code. Specify hexadecimal values in the format '\xHH', where HH can represent any value from 00 to FF. For example, you can specify `BL_COLUMN_DELIMITER='\\x09'` to use the tab character as a delimiter.

## **BL\_COMPRESS=** Data Set Option

Specifies whether to compress data files using the gzip format.

|              |                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                          |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                  |
| Default:     | NO                                                                                                                                                                                                |
| Requirement: | To specify this option, you must first specify <code>BULKLOAD=YES</code> or <code>BULKUNLOAD=YES</code> .                                                                                         |
| Data source: | Amazon Redshift, Microsoft SQL Server, Snowflake                                                                                                                                                  |
| Notes:       | Support for this data set option was added in SAS 9.4M4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7. |
| See:         | <a href="#">BL_COMPRESS= LIBNAME option</a> , <a href="#">BULKLOAD=</a> data set option                                                                                                           |

## Syntax

**BL\_COMPRESS=YES | NO**

## Details

If you have a slow network or very large data files, using `BL_COMPRESS=YES` might improve performance.

## **BL\_CONFIG=** Data Set Option

Specifies the location of the optional configuration file that the SAS communication layer uses to interact with the Amazon S3 tool.

|             |                                                                          |
|-------------|--------------------------------------------------------------------------|
| Valid in:   | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories: | Bulk Loading                                                             |

### Data Set Control

Alias: BL\_CONFIG\_FILE= [Snowflake]  
 Default: none  
 Requirement: To specify this option, you must first specify BULKLOAD=YES.  
 Data source: Amazon Redshift, Snowflake  
 Notes: Support for this data set option was added in SAS 9.4M4.  
           Support for Snowflake was added in the August 2019 release of SAS/ACCESS.  
 See: [BL\\_CONFIG= LIBNAME option](#), [BULKLOAD= data set option](#)  
 Examples:  
       BL\_CONFIG='c:\temp\'  
           [Windows]  
       BL\_CONFIG='/temp/'  
           [UNIX]

## Syntax

**BL\_CONFIG=<path-and-file-name>**

### Required Argument

#### *path-and-file-name*

specifies the host-specific directory path where the optional configuration file for communications with S3 is located.

## Details

The configuration file that you specify can contain options for communication with the Amazon S3 tool. The file format uses name=value syntax with one option per line. For example, you can specify the following S3 options in this file:

```
ssl=yes
keyId=access-key-ID
secret=secret-access-key
region=aws-region
sessionToken=temporary-token
```

For more information, see [information about credentials](#) on the Amazon Web Services website.

---

## BL\_CONTROL\_FIELD\_DELIMITER= Data Set Option

Specifies override of the default delimiter character for separating columns of data during data transfer or retrieval during bulk loading.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                 |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Aliases:     | BL_CONTROL_FIELD_DELIMITER<br>BL_DELIMITER<br>BL_FIELD_DELIMITER                                                                                                                                                                                                                                                                                                                                                                                         |
| Default:     | ,                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Range:       | You can specify any single-character.                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                                                                                                                                                                                                                                             |
| Data source: | SAP HANA                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Note:        | When you specify this option, FIELD DELIMITED BY <i>field-delimiter</i> is added to the SAP HANA IMPORT statement.                                                                                                                                                                                                                                                                                                                                       |
| See:         | <a href="#">BL_CONTROL_QUOTATION_MARK= data set option</a> ,<br><a href="#">BL_CONTROL_RECORD_DELIMITER= data set option</a> , <a href="#">BL_FILE_BADFILE= data set option</a> , <a href="#">BL_FILE_CONTROLFILE= data set option</a> , <a href="#">BL_FILE_DATAFILE= data set option</a> , <a href="#">BL_FILE_DEFAULT_DIR= data set option</a> , <a href="#">BL_FILE_DELETE_DATAFILE= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_CONTROL\_FIELD\_DELIMITER='<field-delimiter>'**

---

## BL\_CONTROL\_QUOTATION\_MARK= Data Set Option

Specifies the quotation character for CSV mode.

|             |                                                                          |
|-------------|--------------------------------------------------------------------------|
| Valid in:   | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories: | Bulk Loading<br>Data Set Control                                         |
| Aliases:    | BL_CONTROL_QUOTATION_MARK<br>BL_QUOTATION<br>BL_QUOTE                    |

Default: " (double quotation mark)

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

See: [BL\\_CONTROL\\_FIELD\\_DELIMITER= data set option](#), [BL\\_CONTROL\\_RECORD\\_DELIMITER= data set option](#), [BL\\_FILE\\_BADFILE= data set option](#), [BL\\_FILE\\_CONTROLFILE= data set option](#), [BL\\_FILE\\_DATAFILE= data set option](#), [BL\\_FILE\\_DEFAULT\\_DIR= data set option](#), [BL\\_FILE\\_DELETE\\_DATAFILE= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_CONTROL\_QUOTATION\_MARK=***quotation-mark*

## BL\_CONTROL\_RECORD\_DELIMITER= Data Set Option

Specifies the single delimiter character to use to separate SAP HANA table records.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Aliases: BL\_CONTROL\_RECORD\_DELIMITER  
BL\_RECORD\_DELIMITER

Default: '\n' (new line)

Range: You can specify any single-character.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

Note: When you specify this option, RECORD DELIMITED BY *record-delimiter* is added to the SAP HANA IMPORT statement.

See: [BL\\_CONTROL\\_FIELD\\_DELIMITER= data set option](#), [BL\\_CONTROL\\_QUOTATION\\_MARK= data set option](#), [BL\\_FILE\\_BADFILE= data set option](#), [BL\\_FILE\\_CONTROLFILE= data set option](#), [BL\\_FILE\\_DATAFILE= data set option](#), [BL\\_FILE\\_DEFAULT\\_DIR= data set option](#), [BL\\_FILE\\_DELETE\\_DATAFILE= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_CONTROL\_RECORD\_DELIMITER='***<record-delimiter>***'**

---

## BL\_CONTROL= Data Set Option

Identifies the file that contains control statements.

|              |                                                                                                                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                     |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                                             |
| Alias:       | FE_EXECNAME [Teradata]                                                                                                                                                                                                                                                                                       |
| Default:     | DBMS-specific                                                                                                                                                                                                                                                                                                |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                                                                                                 |
| Data source: | Oracle, Teradata                                                                                                                                                                                                                                                                                             |
| See:         | <a href="#">BL_DATAFILE= data set option</a> , <a href="#">BL_DELETE_DATAFILE= data set option</a> ,<br><a href="#">BL_DELETE_ONLY_DATAFILE= data set option</a> , <a href="#">BULKLOAD= data set option</a> ,<br><a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a> |

---

## Syntax

**BL\_CONTROL=*path-and-control-file-name* [Oracle]**  
**BL\_CONTROL=*path-and-data-file-name* [Teradata]**

### Syntax Description

***path-and-control-file-name* [Oracle]**

specifies the SQL\*Loader file (where SQLLDR control statements are written) that describe the data to include in bulk loading.

***path-and-data-file-name* [Teradata]**

specifies the name of the control file to generate for extracting data with SAS/ACCESS Interface to Teradata using [FastExport](#) multithreaded Read.

**BL\_<table>\_<unique-ID>.ctl [Oracle, Teradata (UNIX or PC Hosts)]**

the default file name on most platforms, where:

***table***

specifies the table name

***unique-ID***

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates the number.

## Details

### Specifics for Oracle

The Oracle interface creates the control file by using information from the input data and SAS/ACCESS options. The file contains Data Definition Language (DDL) definitions that specify the location of the data and how the data corresponds to the database table. It is used to specify exactly how the loader should interpret the data that you are loading from the DATA file (DAT file). By default, SAS/ACCESS creates a control file in the temporary file directory that is specified by the UTILLOC= system option or in another directory that is based on your default file specifications. If you do not specify this option and a control file does not already exist, a file is created in the temporary file directory or in another directory that is based on your default file specifications. If you do not specify this option and a control file already exists, the Oracle interface reuses the file and replaces the contents with the new control statements.

### Specifics for Teradata

To specify this option, you must first specify DBSCLICEPARM=ALL as a LIBNAME or data set option for threaded Reads. By default, SAS creates a data file in the temporary file directory that is specified by the UTILLOC= system option with a platform-specific name. If you do not specify this option and a control file does not exist, SAS creates a script file in the temporary file directory or in another location that is based on your default file specifications. This file is deleted when the partitioning process is complete.

The script file contains FastExport Language definitions that specify the location of the data and how the data corresponds to the database table. It is used to specify exactly how the FastExport should interpret the data that you are loading from the DATA (.DAT) file. Because the script file that SAS generates for FastExport must contain logon information in clear text, it is recommended that you secure the script file by specifying a directory path that is protected.

### File Naming for Teradata under z/OS

By default under z/OS, the control file name is prefixed with your user ID. This happens automatically without having to specify it. However, you can suppress the addition of your user ID to the file name and add text of your choice instead.

Typically, you would add a security group ID to the beginning of the file name in place of your user ID. When the addition of your user ID is suppressed, you specify the text to add to the beginning of the file name. To prevent adding your user ID to the file name, enable the TD\_RACF environment variable when you start SAS. To set the TD\_RACF environment variable, include the following code in your SAS command:

Alternatively, you can specify TD\_RACF as a SAS option.

---

**Note:** You can use the TD\_RACF environment variable even when you use a security method other than RACF.

---

For more information, see “[Example 2: Under z/OS: Add a Security Group Name to the File Name](#)” on page 398.

---

## Example: Generate Teradata Script Files

This example generates a Teradata script file, C:\protdir\fe.ctl on Windows.

```
DATA test;
SET teralib.mydata(DBSLICEPARM=ALL BL_CONTROL="C:\protdir\fe.ctl");
run;
```

This example generates a Teradata script file, /tmp/fe.ctl, on UNIX.

```
DATA test;
SET teralib.mydata(DBSLICEPARM=ALL BL_CONTROL="/tmp/fe.ctl");
run;
```

This example generates a script file, USERID.SECURE.SCR.CTL, by appending CTL and prepending the user ID.

```
DATA test;
SET teralib.mydata(DBSLICEPARM=ALL BL_CONTROL="SECURE.SCR");
run;
```

---

## BL\_COPY\_LOCATION= Data Set Option

Specifies the directory to which DB2 saves a copy of the loaded data.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES and BL\_RECOVERABLE=YES.

Data source: DB2 under UNIX and PC Hosts

See: [BL\\_RECOVERABLE= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_COPY\_LOCATION=***path-name*

---

## BL\_CPU\_PARALLELISM= Data Set Option

Specifies the number of processes or threads to use when building table objects.

|              |                                                                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                        |
| Default:     | none                                                                                                                                                    |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                            |
| Data source: | DB2 under UNIX and PC Hosts                                                                                                                             |
| See:         | <a href="#">BL_DATA_BUFFER_SIZE= data set option</a> , <a href="#">BL_DISK_PARALLELISM= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_CPU\_PARALLELISM=***number of processes or threads*

### Syntax Description

#### ***number of processes or threads***

specifies the number of processes or threads that the load utility uses to parse, convert, and format data records when building table objects.

---

## Details

This option exploits intrapartition parallelism and significantly improves load performance. It is particularly useful when loading presorted data, because record order in the source data is preserved.

The maximum allowable number is 30. If the value is 0 or has not been specified, the load utility selects an intelligent default. This default is based on the number of available CPUs on the system at run time. If there is insufficient memory to support the specified value, the utility adjusts the value.

When BL\_CPU\_PARALLELISM is greater than 1, the flushing operations are asynchronous, permitting the loader to exploit the CPU. If tables include either LOB or LONG VARCHAR data, parallelism is not supported. The value is set to 1 regardless of the number of system CPUs or the specified value.

Although use of this parameter is not restricted to symmetric multiprocessor (SMP) hardware, you might not obtain any discernible performance benefit from using it in non-SMP environments.

For more information about using BL\_CPU\_PARALLELISM=, see the CPU\_PARALLELISM parameter in the *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*.

---

## BL\_DATA\_BUFFER\_SIZE= Data Set Option

Specifies the total amount of memory to allocate for the bulk-load utility to use as a buffer for transferring data.

|              |                                                                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                               |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                       |
| Default:     | none                                                                                                                                                   |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                           |
| Data source: | DB2 under UNIX and PC Hosts                                                                                                                            |
| See:         | <a href="#">BL_CPU_PARALLELISM= data set option</a> , <a href="#">BL_DISK_PARALLELISM= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_DATA\_BUFFER\_SIZE=*buffer-size***

### Syntax Description

#### *buffer-size*

specifies the total amount of memory that is allocated for the bulk-load utility to use as buffered space for transferring data within the utility. Memory is calculated in 4 KB pages regardless of the degree of parallelism.

---

## Details

If you specify a value that is less than the algorithmic minimum, the minimum required resource is used and no warning is returned. This memory is allocated directly from the utility heap, the size of which you can modify through the util\_heap\_sz database configuration parameter. If you do not specify a value, the utility calculates an intelligent default at run time. This calculated default is based on a percentage of the free space that is available in the utility heap at the time of instantiation of the loader, as well as on some characteristics of the table.

It is recommended that the buffer be several extents in size. An *extent* is the unit of movement for data within DB2, and the extent size can be one or more 4KB pages. The DATA BUFFER parameter is useful when you are working with large objects (LOBs) because it reduces input and output waiting time. The data buffer is allocated from the utility heap. Depending on the amount of storage available on your system, you should consider allocating more memory for use by the DB2 utilities. You can modify the database configuration parameter util\_heap\_sz accordingly. The default value for the Utility Heap Size configuration parameter is 5000 4KB pages. Because load is only one of several utilities that use memory from

the utility heap, it is recommended that no more than 50% of the pages specified by this parameter be made available for the load utility.

For more information about using this option, see the DATA BUFFER parameter in the *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*.

---

## BL\_DATAFILE\_EXISTS= Data Set Option

Lets you load a table from an existing data set.

|              |                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                        |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                |
| Default:     | NO                                                                                                                                              |
| Restriction: | You cannot use this option if you have specified the BL_USE_PIPE= data set option.                                                              |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                    |
| Data source: | Greenplum, HAWQ                                                                                                                                 |
| Notes:       | Support for this data set option was added for SAS 9.4.<br>Support for HAWQ was added in SAS 9.4M3.                                             |
| See:         | <a href="#">BL_USE_PIPE= data set option</a> , <a href="#">BULKLOAD= LIBNAME option on page 171</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_DATAFILE\_EXISTS=YES | NO**

### Optional Arguments

#### **YES**

indicates that the engine opens the specified file in Read mode if the data set already exists.

#### **NO**

indicates that the engine creates the specified file and opens it in Load mode if the data set does not already exist.

---

## BL\_DATAFILE\_PATH= Data Set Option

Specifies a path for creating a flat file for bulk loading.

|             |                                                                          |
|-------------|--------------------------------------------------------------------------|
| Valid in:   | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories: | Bulk Loading<br>Data Set Control                                         |

|              |                                                              |
|--------------|--------------------------------------------------------------|
| Default:     | none                                                         |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES. |
| Data source: | Aster                                                        |
| Note:        | Support for this data set option was added for SAS 9.4.      |
| See:         | <a href="#">BULKLOAD= data set option</a>                    |

## Syntax

**BL\_DATAFILE\_PATH=*path***

### Syntax Description

***path***

specifies that SAS creates one or more temporary tables.

## Details

When you use this option to specify the path, it uses the automatically generated file name where the temporary flat file is created for bulk loading. If you do not specify this option, a file is created in the temporary file directory that is specified by the UTILLOC= system option.

## Example: Specify a Path for a Temporary File

```
libname dblib aster dsn=ncluster uid=user pwd=password dimension=yes;

data dblib.class (BULKLOAD=YES BL_DATAFILE_PATH='C:\temp\' );
    set sashelp.class;
run;
```

## BL\_DATAFILE= Data Set Option

Identifies the file that contains DBMS data for bulk loading.

|              |                                                                                  |
|--------------|----------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing data using SAS/ACCESS software)              |
| Categories:  | Bulk Loading<br>Data Set Control                                                 |
| Default:     | DBMS-specific                                                                    |
| Requirement: | To specify this option, you must first specify BULKEXTRACT= YES or BULKLOAD=YES. |

|              |                                                                                                                                                                                                                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Netezza, Oracle, PostgreSQL, SAP IQ, Yellowbrick                                                                                                                                                                                                                                   |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                                                                                                     |
| See:         | <a href="#">BL_DATAFILE=[Teradata]</a> , <a href="#">BL_CONTROL= data set option</a> , <a href="#">BL_DELETE_DATAFILE= data set option</a> , <a href="#">BL_DELETE_ONLY_DATAFILE= data set option</a> , <a href="#">BL_PROTOCOL= data set option</a> , <a href="#">BL_USE_PIPE= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_DATAFILE=*path-and-data-file-name***

### Syntax Description

***path-and-data-file-name***

specifies the file that contains the rows of data to load or append into a DBMS table during bulk loading. On most platforms, the default file name takes the form `BL_<table>_<unique-ID>.ext`:

***table***

specifies the table name.

***unique-ID***

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates the number.

***ext***

specifies the file extension (.DAT or .IXF) for the data file.

## Details

**DB2 under UNIX and PC Hosts:** The default is the temporary file directory that is specified by the UTILLOC= system option.

**Greenplum, HAWQ:** This option specifies the name of the external file to load. It is meaningful only when BL\_PROTOCOL= is set to gpfdist or file. If you do not specify this option, the file name is generated automatically. When you specify the file name with a full path, the path overrides the value of the GPOLOAD\_HOME environment variable. However, bulk loading might fail if the path does not match the base directory that the gpfdist utility used.

**Netezza:** You can use this option only when BL\_USE\_PIPE=NO. By default, the SAS/ACCESS engine creates a data file from the input SAS data set in the temporary file directory that is specified by the UTILLOC= system option or by using the default file specifications before SAS/ACCESS calls the bulk loader. The data file contains SAS data that is ready to load into the DBMS. By default, the data file is deleted after the load is completed. To override this behavior, specify `BL_DELETE_DATAFILE=NO`.

**Oracle:** The SAS/ACCESS engine creates this data file from the input SAS data set before calling the bulk loader. The data file contains SAS data that is ready to load

into the DBMS. By default, the data file is deleted after the load is completed. To override this behavior, specify BL\_DELETE\_DATAFILE=NO. If you do not specify this option and a data file does not exist, the file is created in the temporary file directory that is specified by the UTILLOC= system option or in another directory that is specified by your default file specifications. If you do not specify this option and a data file already exists, SAS/ACCESS reuses the file, replacing the contents with the new data. SAS/ACCESS Interface to Oracle on z/OS is the exception: The data file is never reused because the interface causes bulk loading to fail instead of reusing a data file.

**SAP IQ:** By default, the SAS/ACCESS engine creates a data file with a .dat file extension in the temporary file directory that is specified by the UTILLOC= system option or in another directory that is specified by your default file specifications. Also, by default, the data file is deleted after the load is completed. To override this behavior, specify BL\_DELETE\_DATAFILE=NO.

## BL\_DATAFILE= Data Set Option: Teradata

Identifies the file that contains control statements.

|              |                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing data using SAS/ACCESS software)                                                                                 |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                    |
| Default:     | creates a MultiLoad script file in the temporary file directory that is specified by the UTILLOC= system option with a platform-specific name       |
| Requirement: | To specify this option, you must first specify MULTILOAD=YES.                                                                                       |
| Data source: | Teradata                                                                                                                                            |
| Note:        | Additional data sources are supported. See the main entry for the BL_DATAFILE= data set option.                                                     |
| See:         | <a href="#">BL_DATAFILE= data set option</a> (main entry), <a href="#">BL_CONTROL= data set option</a> , <a href="#">MULTILOAD= data set option</a> |

## Syntax

**BL\_DATAFILE=*path-and-data-file-name***

### Syntax Description

#### ***path-and-data-file-name***

specifies the name of the control file to generate for loading data with SAS/ACCESS Interface to Teradata using MultiLoad. On Windows and UNIX platforms, the default file name takes the form BL\_<table>\_<unique-ID>.ctl:

#### ***table***

specifies the table name.

***unique-ID***

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates the number.

On z/OS platforms, the default file name takes the form <*unique-ID*>.<*table*>.dat:

***unique-ID***

specifies the user or group ID that owns the file. By default, this is your user ID. However, you can set this to a security group ID with the TD\_RACF environment variable.

***table***

specifies the table name.

## Details

### Overview of BL\_DATAFILE= for Teradata

The file contains MultiLoad Language definitions that specify the location of the data and how the data corresponds to the database table. It specifies exactly how MultiLoad should interpret the data that you are loading. Because the script file that SAS generates for MultiLoad must contain logon information in clear text, you should secure the script file by specifying a protected directory path.

### Specifics for File Naming under z/OS

By default under z/OS, the control file name is prefixed with your user ID. This happens automatically without having to specify it. However, you can suppress the addition of your user ID to the file name and add text of your choice instead.

Typically, you would add a security group ID to the beginning of the file name in place of your user ID. When the addition of your user ID is suppressed, you specify the text to add to the beginning of the file name. To prevent adding your user ID to the file name, enable the TD\_RACF environment variable when you start SAS. To set the TD\_RACF environment variable, include the following code in your SAS command:

Alternatively, you can specify TD\_RACF as a SAS option.

**Note:** You can use the TD\_RACF environment variable even when you use a security method other than RACF.

## Examples

### Example 1: Generate Teradata Script Files

This example generates a Teradata script file, C:\protodir\ml.ct1, on Windows.

```
DATA teralib.test (DBSLICEPARM=ALL BL_DATAFILE="C:\protodir\ml.ct1");
```

```
SET teralib.mydata;
run;
```

This next example generates a Teradata script file, fe.ctl, for FastExport and ml.ctl for MultiLoad.

```
data teralib.test1(MULTILOAD=YES TPT=NO BL_DATAFILE="ml.ctl");
SET teralib.test2(DBSLICEPARM=ALL BL_CONTROL="fe.ctl");
run;
```

## Example 2: Under z/OS: Add a Security Group Name to the File Name

This example shows how to specify that generated file names should include the dbi2 security group rather than the user ID.

```
/* the new env variable / SAS option TD_RACF should be set to YES when invoking
 * SAS in order to test any scenario:
 *
 * sas -set TD_RACF YES -set SYSIN SASIN
 *
 */

libname tera teradata user=dbitest pw=XXXXXX tdpid=tdp0;
proc delete data=tera.multitest ; run;

/* Multiload utility */
data tera.multitest(multiload=yes
                     bl_datafile='dbi2.test.datafile'
                     bl_control='dbi2.test.control');
set sashelp.class;
run;
```

## BL\_DB2CURSOR= Data Set Option

Specifies a string that contains a valid DB2 SELECT statement that points to either local or remote objects (tables or views).

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | none                                                                     |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD=</a> data set option                                |

---

## Syntax

**BL\_DB2CURSOR='SELECT \* from *file-name*'**

---

## Details

You can use it to load DB2 tables directly from other DB2 and objects that are not DB2. However, before you can select data from a remote location, your database administrator must first populate the communication database with the appropriate entries.

---

## BL\_DB2DATACLAS= Data Set Option

Specifies a data class for a new SMS-managed data set.

|              |                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                  |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                                                                                                          |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                                      |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                                                                                                                                                              |
| Data source: | DB2 under z/OS                                                                                                                                                                                                                                                                                                                                                            |
| See:         | <a href="#">BL_DB2IN= data set option</a> , <a href="#">BL_DB2MGMTCLAS= data set option</a> , <a href="#">BL_DB2PRINT= data set option</a> , <a href="#">BL_DB2REC= data set option</a> on page 408, <a href="#">BL_DB2STORCLAS= data set option</a> [contains sample code], <a href="#">BL_DB2UNITCOUNT= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_DB2DATACLAS=*data-class***

---

## Details

This option applies to the control file (BL\_DB2IN= data set option), the input file (BL\_DB2REC= data set option), and the output file (BL\_DB2PRINT= data set option) for the bulk loader. Use this option to specify a data class for a new SMS-managed data set. SMS ignores this option if you specify it for a data set that SMS does not support. If SMS is not installed or active, the operating environment ignores any data class that BL\_DB2DATACLAS= passes. Your site storage administrator specifies the data class names that you can specify when you use this option.

---

## BL\_DB2DEVT\_PERM= Data Set Option

Specifies the unit address or generic device type to use for permanent data sets that the LOAD utility creates—also SYSIN, SYSREC, and SYSPRINT when SAS allocates them.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | SYSDA                                                                    |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_DB2DEVT\_PERM=***unit-specification*

---

## BL\_DB2DEVT\_TEMP= Data Set Option

Specifies the unit address or generic device type to use for temporary data sets that the LOAD utility creates (Pnch, Copy1, Copy2, RCpy1, RCpy2, Work1, Work2).

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | SYSDA                                                                    |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_DB2DEVT\_TEMP=***unit-specification*

---

## BL\_DB2DISC= Data Set Option

Specifies the SYSDISC data set name for the LOAD utility.

|              |                                                                                       |
|--------------|---------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)              |
| Categories:  | Bulk Loading<br>Data Set Control                                                      |
| Default:     | a generated data set name                                                             |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                          |
| Data source: | DB2 under z/OS                                                                        |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">DB2PRINT= data set option</a> |

---

## Syntax

**BL\_DB2DISC=***data-set-name*

---

## Details

The DSNTUTILS procedure with DISP=(NEW,CATLG,CATLG) allocates this option. This option must be the name of a nonexistent data set, except on a RESTART because it would already have been created. The LOAD utility allocates it as DISP=(MOD,CATLG,CATLG) on a RESTART. The default is a generated data set name, which appears in output that is written to the DB2PRINT= data set option location.

---

## BL\_DB2ERR= Data Set Option

Specifies the SYSERR data set name for the LOAD utility.

|              |                                                                                       |
|--------------|---------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)              |
| Categories:  | Bulk Loading<br>Data Set Control                                                      |
| Default:     | a generated data set name                                                             |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                          |
| Data source: | DB2 under z/OS                                                                        |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">DB2PRINT= data set option</a> |

## Syntax

**BL\_DB2ERR=***data-set-name*

---

## Details

The DSNUTILS procedure with DISP=(NEW,CATLG,CATLG) allocates this option. This option must be the name of a nonexistent data set, except on a RESTART because it would already have been created. The LOAD utility allocates it as DISP=(MOD,CATLG,CATLG) on a RESTART. The default is a generated data set name, which appears in output that is written to the DB2PRINT= data set option location.

---

## BL\_DB2IN= Data Set Option

Specifies the SYSIN data set name for the LOAD utility.

|              |                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                               |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                       |
| Default:     | a generated data set name                                                                                                              |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                           |
| Interaction: | In SAS 9.4, the default value for LRECL is 32767. If you are using fixed-length records (RECFM=F), the default value for LRECL is 256. |
| Data source: | DB2 under z/OS                                                                                                                         |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">BL_DB2LDEXT= data set option</a>                                               |

---

## Syntax

**BL\_DB2IN=***data-set-name*

---

## Details

This option is allocated based on the value of BL\_DB2LDEXT=. It is initially allocated as SPACE=(trk,(10,1),rlse) with the default being a generated data set name, which appears in the DB2PRINT output, with these DCB attributes:

DSORG=PS    LRECL=516  
RECFM=VB    BLKSZE=23476

It supports these DCB attributes for existing data sets:

DSORG=PS  
 RECFM=F, FB, FS, FBS, V, VB, VS, or VBS  
 LRECL=any valid value for RECFM  
 BLKSIZE=any valid value for RECFM

## BL\_DB2LDCT1= Data Set Option

Specifies a string in the LOAD utility control statement between LOAD DATA and INTO TABLE.

|              |                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                               |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                       |
| Default:     | none                                                                                                                                                                                   |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                           |
| Data source: | DB2 under z/OS                                                                                                                                                                         |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">BL_DB2LDCT2= data set option</a> , <a href="#">BL_DB2DCT3= data set option</a><br><a href="#">BL_DB2LDEXT= data set option</a> |

## Syntax

`BL_DB2LDCT1='string'`

## Details

This option specifies a string that contains a segment of the Load Utility Control Statement between 'LOAD DATA' and 'INTO TABLE'. Valid control statement options include but are not limited to RESUME, REPLACE, LOG, and ENFORCE.

You can use DB2 bulk-load control options (BL\_DB2LDCT1=, BL\_DB2LDCT2=, and BL\_DB2DCT3=) to specify sections of the control statement, which the engine incorporates into the control statement that it generates. These options have no effect when BL\_DB2LDEXT=USERUN. You can use these options as an alternative to specifying BL\_DB2LDEXT=GENONLY and editing the control statement to include options that the engine cannot generate. In some cases, it is necessary to specify at least one of these options. An example is running the utility on an existing table where you must specify either RESUME or REPLACE.

The LOAD utility requires that the control statement be in uppercase—except for objects such as table or column names, which must match the table. You must specify values for DB2 bulk-load control options using the correct case.

SAS/ACCESS Interface to DB2 under z/OS cannot convert the entire control statement to uppercase because it might contain table or column names that must remain in lowercase.

---

## BL\_DB2LDCT2= Data Set Option

Specifies a string in the LOAD utility control statement between INTO TABLE *table-name* and (*field-specification*).

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | none                                                                     |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_DB2LDCT2='string'**

---

## Details

Valid control statement options include but are not limited to PART, PREFORMAT, RESUME, REPLACE, and WHEN.

---

## BL\_DB2LDCT3= Data Set Option

Specifies a string in the LOAD utility control statement after (*field-specification*).

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | none                                                                     |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_DB2LDCT3='string'**

## Details

This option handles any options that might be specified for this location in later versions of DB2.

---

## BL\_DB2LDEXT= Data Set Option

Specifies the mode of execution for the DB2 LOAD utility.

|              |                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                               |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                       |
| Default:     | GENRUN                                                                                                                                 |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                           |
| Interaction: | In SAS 9.4, the default value for LRECL is 32767. If you are using fixed-length records (RECFM=F), the default value for LRECL is 256. |
| Data source: | DB2 under z/OS                                                                                                                         |
| See:         | <a href="#">BULKLOAD= data set option</a>                                                                                              |

---

## Syntax

**BL\_DB2LDEXT=GENRUN | GENONLY | USERUN**

### Syntax Description

#### **GENRUN**

generates the control (SYSIN) file and the data (SYSREC) file, and runs the utility with them.

#### **GENONLY**

generates the control (SYSIN) file and the data (SYSREC) file but does not run the utility. Use this method when you need to edit the control file or to verify the generated control statement or data before you run the utility.

#### **USERUN**

uses existing control and data files, and runs the utility with them. Existing files can be from a previous run or from previously run batch utility jobs. Use this method when you restart a previously stopped run of the utility.

All valid data sets that the utility accepts are supported when BL\_DB2LDEXT=USERUN. However, syntax errors from the utility can occur because no parsing is done when reading in the SYSIN data set. Specifically, neither embedded comments (beginning with a double hyphen, '--') nor columns 73 through 80 of RECFM=FB LRECL=80 data sets are stripped from the control statement. The solution is to remove embedded comments and columns 73 through 80 of RECFM=FB LRECL=80 data sets from the data set. However, this

is not an issue when you use engine-generated SYSIN data sets because they are RECFM=VB and therefore have no embedded comments.

---

## Details

This option specifies the mode of execution for the DB2 LOAD utility, which involves creating data sets that the utility needs and to call the utility.

---

## BL\_DB2MGMTCLAS= Data Set Option

Specifies a management class for a new SMS-managed data set.

|              |                                                                                                                                                                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                              |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                      |
| Default:     | none                                                                                                                                                                                                                                                                                  |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                                                                          |
| Data source: | DB2 under z/OS                                                                                                                                                                                                                                                                        |
| See:         | <a href="#">BL_DB2DATACLAS= data set option</a> , <a href="#">BL_DB2REC=</a> , <a href="#">BL_DB2PRINT=</a> ,<br><a href="#">BL_DB2STORCLAS= data set option</a> [contains sample code], <a href="#">BL_DB2UNITCOUNT= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_DB2MGMTCLAS=***management-class*

---

## Details

This option applies to the control file (BL\_DB2IN=), the input file (BL\_DB2REC=), and the output file (BL\_DB2PRINT=) for the bulk loader. Use this option to specify a management class for a new SMS-managed data set. If SMS is not installed or active, the operating environment ignores any management class that BL\_DB2MGMTCLAS= passes. Your site storage administrator specifies the management class names that you can specify when you use this option.

---

## BL\_DB2MAP= Data Set Option

Specifies the SYSMAP data set name for the LOAD utility.

|           |                                                                          |
|-----------|--------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
|-----------|--------------------------------------------------------------------------|

Categories: Bulk Loading  
 Data Set Control

Default: a generated data set name

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under z/OS

See: [BULKLOAD= data set option](#)

## Syntax

**BL\_DB2MAP=***data-set-name*

## Details

The DSNUTILS procedure with DISP=(NEW,CATLG,CATLG) allocates this option. This option must be the name of a nonexistent data set, except on a RESTART because it would already have been created. The LOAD utility allocates it as DISP=(MOD,CATLG,CATLG) on a RESTART. The default is a generated data set name, which appears in output that is written to the DB2PRINT location.

## BL\_DB2PRINT= Data Set Option

Specifies the SYSPRINT data set name for the LOAD utility.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
 Data Set Control

Default: a generated data set name

Requirement: To specify this option, you must first specify BULKLOAD=YES. You must also specify BL\_DB2PRNLOG=YES so that you can see the generated data set name in the SAS log.

Interaction: In SAS 9.4, the default value for LRECL is 32767. If you are using fixed-length records (RECFM=F), the default value for LRECL is 256.

Data source: DB2 under z/OS

See: [BL\\_DB2PRNLOG= data set option](#) [for generated data set name in the SAS log], [BULKLOAD= data set option](#)

## Syntax

**BL\_DB2PRINT=***data-set-name*

## Details

This option is allocated with DISP=(NEW,CATLG,DELETE) and SPACE=(trk,(10,1),rlse). The default is a generated data set name, which appears in the DB2PRINT DSN, with these DCB attributes:

|           |                   |
|-----------|-------------------|
| DSORG=PS  | LRECL=258         |
| RECFM=VBA | BLKSIZE=262–32767 |

---

## BL\_DB2PRNLOG= Data Set Option

Determines whether to write SYSPRINT output to the SAS log.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | YES                                                                      |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD= data set option, DB2 under z/OS Bulk Loading</a>   |

---

## Syntax

**BL\_DB2PRNLOG=YES | NO**

### Syntax Description

#### **YES**

specifies that SYSPRINT output is written to the SAS log.

#### **NO**

specifies that SYSPRINT output is not written to the SAS log.

---

## BL\_DB2REC= Data Set Option

Specifies the SYSREC data set name for the LOAD utility.

|             |                                                                          |
|-------------|--------------------------------------------------------------------------|
| Valid in:   | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories: | Bulk Loading<br>Data Set Control                                         |
| Default:    | a generated data set name                                                |

|              |                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                            |
| Interaction: | In SAS 9.4, the default value for LRECL is 32767. If you are using fixed-length records (RECFM=F), the default value for LRECL is 256.  |
| Data source: | DB2 under z/OS                                                                                                                          |
| See:         | <a href="#">BL_DB2LDEXT= data set option</a> , <a href="#">BL_DB2RECSP= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_DB2REC=***data-set-name*

## Details

The value of BL\_DB2LDEXT= allocates this option. It is initially allocated as SPACE=(cyl,(BL\_DB2RECSP, 10%(BL\_DB2RECSP)),rlse). The default is a generated data set name, which appears in output that is written to the DB2PRINT data set name. It supports these DCB attributes for existing data sets:

|          |                                   |
|----------|-----------------------------------|
| DSORG=PS | LRECL=any valid value for RECFM   |
| RECFM=FB | BLKSIZE=any valid value for RECFM |

## BL\_DB2RECSP= Data Set Option

Determines the number of cylinders to specify as the primary allocation for the SYSREC data set when it is created.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | 10                                                                       |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

## Syntax

**BL\_DB2RECSP=***primary-allocation*

## Details

The secondary allocation is 10% of the primary allocation.

---

## BL\_DB2RSTRT= Data Set Option

Tells the LOAD utility whether the current load is a restart and, if so, indicates where to begin.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: NO

Requirements: To specify this option, you must first specify BULKLOAD=YES.

When you specify a value other than NO for BL\_DB2RSTRT=, you must also specify and BL\_DB2LDEXT=USERUNBL\_DB2TBLXST=YES.

Data source: DB2 under z/OS

See: [BL\\_DB2TBLXST= data set option](#), [BL\\_DB2LDEXT= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_DB2RSTRT=NO | CURRENT | PHASE**

### Syntax Description

#### NO

specifies a new run (not restart) of the LOAD utility.

#### CURRENT

specifies to restart at the last commit point.

#### PHASE

specifies to restart at the beginning of the current phase.

---

## BL\_DB2SPC\_PERM= Data Set Option

Determines the number of cylinders to specify as the primary allocation for permanent data sets that the LOAD utility creates.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: 10

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under z/OS

See: [BULKLOAD= data set option](#)

## Syntax

**BL\_DB2SPC\_PERM=***primary-allocation*

## Details

Permanent data sets are Disc, Maps, and Err. The DSNUTILS procedure controls the secondary allocation, which is 10% of the primary allocation.

## BL\_DB2SPC\_TEMP= Data Set Option

Determines the number of cylinders to specify as the primary allocation for temporary data sets that the LOAD utility creates.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: 10

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under z/OS

See: [BULKLOAD= data set option](#), [DB2 under z/OS Bulk Loading](#)

## Syntax

**BL\_DB2SPC\_TEMP=***primary-allocation*

## BL\_DB2STORCLAS= Data Set Option

Specifies a storage class for a new SMS-managed data set.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: none

|              |                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                                                                                                                                                                         |
| Data source: | DB2 under z/OS                                                                                                                                                                                                                                                                                                                                                                       |
| See:         | <a href="#">BL_DB2DATACLAS=</a> data set option, <a href="#">BL_DB2DEVT_PERM=</a> data set option, <a href="#">BL_DB2IN=</a> data set option, <a href="#">BL_DB2MGMTCLAS=</a> data set option, <a href="#">BL_DB2PRINT=</a> data set option, <a href="#">BL_DB2REC=</a> data set option, <a href="#">BL_DB2UNITCOUNT=</a> data set option, <a href="#">BULKLOAD=</a> data set option |

---

## Syntax

**BL\_DB2STORCLAS=***storage-class*

---

## Details

A storage class contains the attributes that identify a storage service level that SMS uses for storage of the data set. It replaces any storage attributes that you specify in [BL\\_DB2DEVT\\_PERM=](#).

This option applies to the control file (BL\_DB2IN), the input file (BL\_DB2REC=), and the output file (BL\_DB2PRINT=) for the bulk loader. Use this option to specify a management class for a new SMS-managed data set. If SMS is not installed or active, the operating environment ignores any storage class that [BL\\_DB2MGMTCLAS=](#) passes. Your site storage administrator specifies the storage class names that you can specify when you use this option.

---

## Example: Generate SMS-Managed Control and Data Files

This example generates SMS-managed control and data files. It does not create the table, and you do not need to run the utility to load it.

```
libname db2lib db2 ssid=db2a;
data db2lib.customers (bulkload=yes
    bl_db2ldext=genonly
    bl_db2in='myusr1.sysin'
    bl_db2rec=myusr1.sysrec'
    bl_db2tblxst=yes
    bl_db2ldct1='REPLACE'
    bl_db2dataclas='STD'
    bl_db2mgmtclas='STD'
    bl_db2storclas='STD');
set work.customers;
run;
```

---

## BL\_DB2TBLXST= Data Set Option

Indicates whether the LOAD utility runs against an existing table.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: NO

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under z/OS

See: [BL\\_DB2LDEXT= data set option](#), [BL\\_DB2LDICT1= data set option](#), [BL\\_DB2LDICT2 data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_DB2TBLXST=YES | NO**

### Syntax Description

#### **YES**

specifies that the LOAD utility runs against an existing table. This is *not* a replacement operation. (See “[Details](#)” on page 413.)

#### **NO**

specifies that the LOAD utility does not run against an existing table.

---

## Details

SAS/ACCESS does not currently support table replacement. You cannot simply create a new copy of an existing table to replace the original table. Instead, you must delete the table and then create a new version of it.

The DB2 LOAD utility does not create tables—it loads data into existing tables. The DB2 under z/OS interface creates a table before loading data into it—whether you use SQL INSERT statements or start the LOAD utility.

You might want to start the utility for an existing table that the DB2 engine did not create. If so, specify BL\_DB2TBLXST=YES to tell the engine that the table already exists. When BL\_DB2TBLXST=YES, the engine neither verifies that the table does not already exist, which eliminates the NO REPLACE error, nor creates the table. BULKLOAD= is not valid for update opening of tables, which includes appending to an existing table. Therefore, to accomplish appending, use either BL\_DB2TBLXST= with an output open (normally creates the table) or the LOAD utility against a previously created table. You can also use BL\_DB2TBLXST= with BL\_DB2LDEXT=GENONLY if the table does not yet exist and you do not want to

create or load it yet. In this case the control and data files are generated but the table is neither created nor loaded.

Because the table might be empty or might contain rows, specify the appropriate LOAD utility control statement values for REPLACE, RESUME, or both by using BL\_DB2LDICT1, BL\_DB2LDICT2, or both.

The data to be loaded into the existing table must match the table column types. The engine does not try to verify input data with the table definition. The LOAC utility flags any incompatible differences.

## BL\_DB2UNITCOUNT= Data Set Option

Specifies the number of volumes on which data sets can be extended.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

Restriction: An error is returned if you specify a value for this option that exceeds the maximum number of volumes for the unit.

Requirements: To specify this option, you must first specify BULKLOAD=YES.

This option applies only to the input file (BL\_DB2REC data set). This is the file that must be loaded into the DB2 table.

Data source: DB2 under z/OS

See: [BL\\_DB2DATACLAS= data set option](#), [BL\\_DB2DEVT\\_PERM= data set option](#), [BL\\_DB2STORCLAS= data set option](#) [contains sample code], [BL\\_DB2STORCLAS= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_DB2UNITCOUNT=***number-of-volumes*

### Syntax Description

#### ***number-of-volumes***

specifies the number of volumes across which data sets can be extended. It must be an integer between 1 and 59. This option is ignored if the value is greater than 59. See the details in this section.

## Details

You must specify an integer from 1–59 as a value for this option. This option is ignored if the value is greater than 59. However, the value depends on the unit name in BL\_DB2DEVT\_PERM=. At the operating environment level an association

exists that specifies the maximum number of volumes for a unit name. Ask your storage administrator for this number.

The data class determines whether SMS-managed data sets can be extended on multiple volumes. When you specify both BL\_DB2DATACLAS= and BL\_DB2UNITCOUNT=, BL\_DB2UNITCOUNT= overrides the unit count values for the data class.

---

## BL\_DB2UTID= Data Set Option

Specifies a unique identifier for a given run of the DB2 LOAD utility.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | user ID and second level DSN qualifier                                   |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_DB2UTID=*utility-ID***

### Syntax Description

***utility-ID***

a character string up to 16 bytes long.

---

## Details

By default, the value for this option is the user ID concatenated with the second-level data set name qualifier. The generated ID appears in output that is written to the DB2PRINT data set name. This name generation makes it easy to associate all information for each utility execution and to separate it from other executions.

---

## BL\_DBNAME= Data Set Option

Specifies the database name to use for bulk loading.

|             |                                                                          |
|-------------|--------------------------------------------------------------------------|
| Valid in:   | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories: | Bulk Loading                                                             |

|               |                                                                                                                                 |
|---------------|---------------------------------------------------------------------------------------------------------------------------------|
|               | Data Set Control                                                                                                                |
| Default:      | none                                                                                                                            |
| Requirements: | To specify this option, you must first specify BULKLOAD=YES.<br>You must enclose the database name in quotation marks.          |
| Data source:  | Aster                                                                                                                           |
| See:          | <a href="#">BL_HOST= data set option</a> , <a href="#">BL_PATH= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_DBNAME='database-name'**

### Syntax Description

#### ***database-name***

specifies the database name to use for bulk loading.

## Details

Use this option to pass the database name to the DBMS bulk-load facility.

## BL\_DEFAULT\_DIR= Data Set Option

Specifies where bulk loading creates all intermediate files.

|               |                                                                                                                                                                                                                                                 |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                        |
| Categories:   | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                |
| Defaults:     | <b><i>database-name</i> [Oracle]</b><br><i>temporary-file-directory</i> that is specified by the UTILLOC= system option [Amazon Redshift, Google BigQuery, Microsoft SQL Server, MySQL, Netezza, Oracle, PostgreSQL, Snowflake, Yellowbrick]    |
| Requirements: | To specify this option, you must first specify BULKLOAD=YES.<br>This option must end with a backslash on Windows or forward slash on UNIX.                                                                                                      |
| Supports:     | CTL, DAT, LOG, BAD, DSC intermediate bulk-load files (Oracle) or CTL and DAT intermediate bulk-load files (PostgreSQL)                                                                                                                          |
| Data source:  | Amazon Redshift, Google BigQuery, Microsoft SQL Server, MySQL, Netezza, Oracle, PostgreSQL, Snowflake, Yellowbrick                                                                                                                              |
| Notes:        | Support for Amazon Redshift was added in SAS 9.4M4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |

See: [BULKLOAD= data set option](#)

Examples: `BL_DEFAULT_DIR='c:\temp\'`

[Windows]

`BL_DEFAULT_DIR='/temp/'`

[UNIX]

## Syntax

`BL_DEFAULT_DIR='host-specific-directory-path'`

### Required Argument

***host-specific-directory-path***

specifies the host-specific directory path where intermediate bulk-load files are created.

## Details

The value that you specify for this option is prepended to the file name. Be sure to provide the complete, host-specific directory path, including the file and directory separator character to accommodate all platforms.

*Microsoft SQL Server:* This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with a Microsoft Azure Data Lake Storage Gen2 storage account. For more information, “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

## Example: Create All Files in a Temporary Directory

In this example, bulk loading creates all related files in the `c:\temp` directory on a Windows system.

```
data x.test (bulkload=yes BL_DEFAULT_DIR="c:\temp\"  
bl_delete_files=no);  
c1=1;  
run;
```

## BL\_DELETE\_DATAFILE= Data Set Option

Specifies whether to delete only the data file or all files that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                       |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                                                                                                                                               |
| Alias:       | BL_DELETE_FILES= [Oracle]                                                                                                                                                                                                                                                                                                                                                                                      |
| Default:     | YES                                                                                                                                                                                                                                                                                                                                                                                                            |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                                                                                                                                                                                                   |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, Oracle, PostgreSQL, SAP IQ, Snowflake, Yellowbrick                                                                                                                                                                                                                               |
| Notes:       | Support for PostgreSQL was added for SAS 9.4.<br>Support for Impala was added in SAS 9.4M2.<br>Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in SAS 9.4M4.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">BL_CONTROL= data set option</a> , <a href="#">BL_DATAFILE= data set option</a> ,<br><a href="#">BL_DELETE_DATAFILE= LIBNAME option</a> , <a href="#">BL_DELETE_ONLY_DATAFILE= data set option</a> , <a href="#">BL_USE_MANIFEST= data set option</a> , <a href="#">BL_USE_PIPE= data set option</a> ,<br><a href="#">BULKLOAD= data set option</a>                                                 |

---

## Syntax

**BL\_DELETE\_DATAFILE=YES | NO**

### Syntax Description

#### YES

deletes all (data, control, and log) files that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

#### NO

does not delete these files.

---

## Details

*Amazon Redshift:* Setting BL\_DELETE\_DATAFILES=YES deletes data files that are created during the bulk-loading process. If BL\_USE\_MANIFEST=YES, then manifest files are deleted as well. Files are deleted from the local machine and from the S3 bucket.

*DB2 under UNIX and PC Hosts:* Setting BL\_DELETE\_DATAFILE=YES deletes only the temporary data file that SAS/ACCESS creates after the load completes.

*Greenplum, HAWQ:* When BL\_DELETE\_DATAFILE=YES, the external data file is deleted after the load completes.

*Netezza:* You can use this option only when BL\_USE\_PIPE=NO.

*Oracle, PostgreSQL:* When BL\_DELETE\_DATAFILE=YES, all files (DAT, CTL, and LOG) are deleted.

## Examples

### Example 1: Delete All Files

In this example, the default is YES, so all files are deleted.

```
libname x oracle user=myusr1 pw=mypwd1 path=mypath;
proc datasets library=x;
  delete test1; run;
  data x.test1 ( bulkload=yes );
    c1=1;
  run;
  x dir BL_TEST1*.*;
```

### Example 2: Retain All Files

No files are deleted in this example.

```
libname x oracle user=myusr1 pw=mypwd1 path=mypath;
proc datasets library=x;
  delete test2; run;
  data x.test2 ( bulkload=yes bl_delete_files=no );
    c1=1;
  run;
  x dir BL_TEST2*.*;
```

## BL\_DELETE\_ONLY\_DATAFILE= Data Set Option

Specifies whether to delete the data file that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

|              |                                                                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                        |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                |
| Default:     | none                                                                                                                                                                                            |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                    |
| Data source: | Oracle                                                                                                                                                                                          |
| See:         | <a href="#">BL_CONTROL= data set option</a> , <a href="#">BL_DATAFILE= data set option</a> ,<br><a href="#">BL_DELETE_DATAFILE= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_DELETE\_ONLY\_DATAFILE=YES | NO**

### Syntax Description

#### YES

deletes only the data file that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

#### NO

does not delete the data file.

## Details

Specifying this option overrides the [BL\\_DELETE\\_DATAFILE=](#) option.

## Examples

### Example 1: Delete Only the Control and Log Files

`BL_DELETE_DATAFILE=YES` is the default in this example, so only the control and log files are deleted.

```
proc datasets library=x;
  delete test3;run;
data x.test3 ( bulkload=yes bl_delete_only_datafile=no );
c1=1;
run;
x dir BL_TEST3*.*;
```

### Example 2: Retain All Files

Both options are set to NO in this example, so no files are deleted.

```
proc datasets library=x;
  delete test4;run;
data x.test4 ( bulkload=yes bl_delete_only_datafile=no bl_delete_files=NO );
c1=1;
run;
x dir BL_TEST4*.*;
```

### Example 3: Delete Only the Data File

Only the data file is deleted in this example.

```
proc datasets library=x;
  delete test5;run;
```

```
data x.test5 ( bulkload=yes bl_delete_only_datafile=YES );
c1=1;
run;
x dir BL_TEST5*.*;
```

The same is true in this example.

```
proc datasets library=x;
  delete test6;run;
run;
data x.test6 ( bulkload=yes bl_delete_only_datafile=YES  bl_delete_files=NO );
c1=1;
run;
x dir BL_TEST6*.*;
```

## BL\_DELIMITER= Data Set Option

Specifies override of the default delimiter character for separating columns of data during data transfer or retrieval during bulk loading or unloading.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Alias: DELIM=, DELIMIT= [Hadoop]

Default: DBMS-specific

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Amazon Redshift, Aster, Google BigQuery, Greenplum, Hadoop, HAWQ, Microsoft SQL Server, Netezza, PostgreSQL, Snowflake, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in SAS 9.4M4.

Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Microsoft SQL Server was added in SAS 9.4M7.

Support for Yellowbrick was added in SAS 9.4M7.

Tip: You can also use this option with the BULKUNLOAD= option, if your interface supports that option.

See: [BL\\_DATAFILE= data set option](#), [BL\\_DELETE\\_DATAFILE= data set option](#), [BL\\_DELIMITER= LIBNAME option](#), [BL\\_FORCE\\_NOT\\_NULL= data set option](#), [BL\\_FORMAT= data set option](#), [BL\\_NULL= data set option](#), [BL\\_OPTIONS= data set option](#), [BL\\_QUOTE= data set option](#), [BL\\_USE\\_ESCAPE= data set option](#), [BL\\_USE\\_PIPE= data set option](#), [BULKLOAD= data set option](#), [BULKUNLOAD= LIBNAME option](#), [BULKUNLOAD= data set option](#)

## Syntax

**BL\_DELIMITER='any-single-character'**

## Details

### Overview

Here is when you might want to use this option:

- to override the default delimiter character that the interface uses to separate columns of data that are transferred to or retrieved from the DBMS during bulk loading. For Netezza, this also applies to bulk unloading.
- if your character data contains the default delimiter character, to avoid any problems while parsing the data stream

You must ensure that the characters that are assigned to `BL_DELIMITER=` and `BL_QUOTE=` are different.

### DBMS Specifics

*Amazon Redshift*: The default is the bell character (ASCII 0x07).

*Aster*: The default is /t (the tab character).

*Google BigQuery*: The default is the bell character (ASCII 0x07).

*Greenplum*: The default is the pipe symbol (|).

*Hadoop*: The default is \001 (Ctrl-A). To change the default delimiter, specify a value as either a single character or three-digit decimal ASCII value between 001 and 127. The value represents the ASCII value of the delimiter that you want to use. You cannot use other typical SAS or UNIX formats such as '\001', 0x01 or '01'x because these do not work. Also, for such procedures as APPEND, SQL, or INSERT, the existing delimiter of the base table—the one being appended to—overrides any specified value for the `DELIMITER=` option. Otherwise, data corruption would result because the original and appended parts of the resulting table would use different delimiters.

*HAWQ*: The default is the pipe symbol (|).

*Microsoft SQL Server*: The default is the bell character (ASCII 0x07).

*Netezza*: You can use any 7-bit ASCII character as a delimiter. The default is the pipe symbol (|). To use a printable ASCII character, enclose it in quotation marks (for example, `BL_DELIMITER="|"`). However, to use an extended character, use the three-digit decimal number representation of the ASCII character for this option. For example, set `BL_DELIMITER=202` to use ASCII character 202 as a delimiter. You must specify decimal number delimiters as three digits even if the first two digits would be zero. For example, specify `BL_DELIMITER=003`, not `BL_DELIMITER=3` or `BL_DELIMITER=03`.

*PostgreSQL*: The default is the pipe character (|).

## Examples

### Example 1: Override the Default Pipe Delimiter

Data in Testdel data set contains the pipe character. Use the BL\_DELIMITER= data set option to override the default ‘|’ pipe delimiter in PROC APPEND.

```
data work.testdel;
    col1='my|data';col2=12;
run;

/* Use a comma to delimit data */
proc append base=netlib.mydat (BULKLOAD=YES BL_DELIMITER=',')
    data=work.testdel;
run;
```

### Example 2: Override the Default Hadoop Delimiter

```
data db.joeapp (delim=007); set db.JoeTable2; run;
data db.joeapp (delim="127"); set db.JoeTable2; run;
data db.joeapp (delimit=#); set db.JoeTable2; run;
data db.joeapp (delimit="#"); set db.JoeTable2; run;

proc sql;
    create table db.joeapp (delim='#') as select * from db.JoeTable2;
quit;
```

---

## BL\_DIRECT\_PATH= Data Set Option

Specifies the Oracle SQL\*Loader DIRECT option.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | YES                                                                      |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | Oracle                                                                   |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_DIRECT\_PATH=YES | NO**

## Syntax Description

### YES

specifies the Oracle SQL\*Loader option DIRECT to TRUE, letting the SQL\*Loader use Direct Path Load to insert rows into a table.

### NO

specifies the Oracle SQL\*Loader option DIRECT to FALSE, letting the SQL\*Loader use Conventional Path Load to insert rows into a table.

## Details

Conventional Path Load reads in multiple data records and places them in a binary array. When the array is full, it is passed to Oracle for insertion and Oracle uses the SQL interface with the array option.

Direct Path Load creates data blocks that are already in the Oracle database block format. Blocks are then written directly into the database. This method is significantly faster, but there are restrictions. For more information about the SQL\*Loader Direct and Conventional Path loads, see your Oracle utilities documentation for SQL\*Loader.

## BL\_DISCARDFILE= Data Set Option

Identifies the file that contains records that were filtered from bulk loading because they did not match the criteria as specified in the CONTROL file.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: Unless you specify the BL\_DEFAULT\_DIR= data set option, the BL\_DISCARDFILE= option creates a file in the temporary file directory that is specified by the UTILLOC= system option (or in another directory that is based on your default file specifications).

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Oracle

See: [BL\\_BADFILE= data set option](#) [to specify the name and location of the file that contains rejected rows], [BULKLOAD= data set option](#)

## Syntax

**BL\_DISCARDFILE=***path-and-discard-file-name*

## Syntax Description

***path-and-discard-file-name***

an SQL\*Loader discard file containing rows that did not meet the specified criteria. On most platforms, the default file name takes the form  
BL\_<table>\_<unique-ID>.dsc:

***table***

specifies the table name

***unique-ID***

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates the number.

---

## Details

The SQL\*Loader creates a file of discarded rows whenever rows are discarded during the load. If you do not explicitly specify a discard file, SQL\*Loader creates one in the default UTILLOC location. You can tell SQL\*Loader to use a discard file of your choice. In this case, if SQL\*Loader generates discarded rows, the contents of the file are overwritten.

On most operating systems, the discard file has the same format as the data file, so discarded records can be loaded after corrections are made.

**Operating Environment Information:** On z/OS operating systems, the discard file is created with default DCB attributes. For information about how to overcome such a case, see the section about SQL\*Loader file attributes in the SQL\*Loader chapter in the Oracle user's guide for z/OS.

---

## BL\_DISK\_PARALLELISM= Data Set Option

Specifies the number of processes or threads to use when writing data to disk.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under UNIX and PC Hosts

See: [BL\\_CPU\\_PARALLELISM= data set option](#), [BL\\_DATA\\_BUFFER\\_SIZE= data set option](#), [BULKLOAD= set option](#)

---

## Syntax

**BL\_DISK\_PARALLELISM=***number-of-processes-or-threads*

## Syntax Description

### **number of processes or threads**

specifies the number of processes or threads that the load utility uses to write data records to the tablespace containers.

## Details

This option exploits the available containers when it loads data and significantly improves load performance.

The maximum number that is allowed is the greater of 50 or four times the BL\_CPU\_PARALLELISM value, which the load utility actually uses. By default, BL\_DISK\_PARALLELISM is equal to the sum of the tablespace containers on all tablespaces that contain objects for the table that is being loaded. However, this value cannot exceed the maximum allowed value.

If you do not specify a value, the utility selects an intelligent default that is based on the number of tablespace containers and the characteristics of the table.

For more information about using this option, see the DISK\_PARALLELISM parameter in the *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*.

## BL\_DNSSUFFIX= Data Set Option

Specifies the network host name where the storage system resides.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Access

Default: LIBNAME option value

Interaction: This option is used with the BL\_ACCOUNTNAME=,BL\_FILESYSTEM=, and BL\_FOLDER= options to specify the external data location.

Data source: Microsoft SQL Server

Note: Support for this data set option was added in SAS 9.4M7.

See: [BL\\_ACCOUNTNAME= data set option](#), [BL\\_DNSSUFFIX= LIBNAME option](#),  
[BL\\_FILESYSTEM= data set option](#), [BL\\_FOLDER= data set option](#)

## Syntax

**BL\_DNSSUFFIX="network-storage-host-name"**

## Required Argument

### ***network-storage-host-name***

specifies the network host name where the storage system resides. It is referred to as a suffix because the value is typically appended to the end of the storage account name.

---

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

The external data location is generated using this option and the BL\_ACCOUNTNAME=, BL\_FILESYSTEM=, and BL\_FOLDER= data set options. These values are combined to define the URL to access an Azure Data Lake Storage Gen2 location:

`https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>`

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

---

## BL\_ENCKEY= Data Set Option

Specifies the name of the encryption key to use when you access an Amazon S3 environment.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Access

Aliases: BL\_ENC\_KEY=

BL\_ENCRYPTION\_KEY=

Default: LIBNAME option value

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Amazon Redshift, Snowflake

Notes: Support for this data set option was added in SAS 9.4M6.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Tip: You can also use this option with the BULKUNLOAD= option.

See: [BL\\_ENCKEY= LIBNAME option](#), [BULKLOAD= data set option](#), [BULKUNLOAD= data set option](#), [S3 procedure documentation](#)

---

## Syntax

**BL\_ENCKEY=***encryption-key-name*

## Required Argument

### ***encryption-key-name***

specifies the name that is associated with an encryption key. The name matches a name that was assigned to an encryption key by using the S3 procedure.

---

## Details

When you specify the BL\_ENCKEY= option, you enable server-side encryption for all communication with an Amazon S3 environment.

You use the S3 procedure to register an encryption key and assign a corresponding name. This is the name that you provide as the value for the BL\_ENCKEY= option. For more information, see the ENCKEY statement in the “[S3 Procedure](#)” in *Base SAS Procedures Guide*.

---

## BL\_ENCODING= Data Set Option

Specifies the character set encoding to use for the external table.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | DEFAULT                                                                  |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | Greenplum, HAWQ                                                          |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                                 |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_ENCODING=***character-set-encoding*

### Syntax Description

#### ***character-set-encoding***

specifies the character set encoding to use for the external table. Specify a string constant (such as 'SQL\_ASCII'), an integer-encoding number, or DEFAULT to use the default client encoding.

---

# BL\_ESCAPE= Data Set Option

Specifies the single character to use for C escape sequences.

|              |                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                 |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                         |
| Default:     | \ (backslash)                                                                                                                                            |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                             |
| Interaction: | <i>PostgreSQL</i> : This option only applies to CSV files. Therefore, you must also set BLFORMAT=CSV.                                                    |
| Data source: | Aster, Greenplum, HAWQ, PostgreSQL, Yellowbrick                                                                                                          |
| Notes:       | Support for this data set option was added in SAS 9.4M1.<br>Support for Aster was added in SAS 9.4M2.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">BL_FORMAT= data set option</a> , <a href="#">BULKLOAD= data set option</a>                                                                   |

---

## Syntax

`BL_ESCAPE='any-single-character' | 'OFF'`

---

## Details

Use this option to specify the single character to use for character escape sequences. These can be \n, \t, or \001 (start of heading). It can also be for escape data characters that might otherwise be used as row or column delimiters. Be sure to choose one that is not used anywhere in your actual column data.

Although the default is \ (backslash), you can specify any other character. You can also specify 'OFF' to disable the use of escape characters. This is very useful for web log data that contains numerous embedded backslashes that are not intended as escape characters.

For octal codes, specify a backslash followed by a three-digit code that represents your escape character, such as \041 ('!'). In SAS 9.4M3, octal notation that uses a preceding 'E', such as E'24', is not supported. Some common escape characters and their corresponding octal codes are listed in the following table.

*Table 13.1 Octal Codes for Common Escape Characters*

| Escape Character   | Octal Code |
|--------------------|------------|
| (start of heading) | \001       |

| Escape Character   | Octal Code |
|--------------------|------------|
| (device control 4) | \024       |
| !                  | \041       |
| /                  | \057       |
| ^                  | \136       |
| ~                  | \176       |

---

## BL\_EXCEPTION= Data Set Option

Specifies the exception table into which rows in error are copied.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

Requirements: To specify this option, you must set BULKLOAD=YES.

To specify this option, you must set BULKLOAD=YES, and you must specify values for BL\_REJECT\_TYPE= and BL\_REJECT\_LIMIT= data set options.

Data source: DB2 under UNIX and PC Hosts, Greenplum, HAWQ

Note: Support for HAWQ was added in SAS 9.4M3.

See: [BL\\_REJECT\\_LIMIT= data set option](#), [BL\\_REJECT\\_TYPE= data set option](#), [BULKLOAD= data set option](#), [Capturing Bulk-Load Statistics into Macro Variables](#)

---

## Syntax

**BL\_EXCEPTION=***exception-table-name*

### Syntax Description

***exception table-name***

specifies the exception table into which rows in error are copied.

## Details

**DB2 under UNIX and PC Hosts:** Any row that is in violation of a unique index or a primary key index is copied. DATALINK exceptions are also captured in the exception table. If you specify an unqualified table name, the table is qualified with the CURRENT SCHEMA. Information that is written to the exception table is not written to the dump file. In a partitioned database environment, you must specify an exception table for those partitions on which the loading table is stored. However, the dump file contains rows that cannot be loaded because they are not valid or contain syntax errors.

For more information about using this option, see the FOR EXCEPTION parameter in *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*. For more information about the load exception table, see the load exception table topics in *IBM DB2 Universal Database Data Movement Utilities Guide and Reference* and *IBM DB2 Universal Database SQL Reference, Volume 1*.

**Greenplum, HAWQ:** Formatting errors are logged when running in single-row, error-isolation mode. You can then examine this error table to determine whether any error rows were not loaded. The specified error table is used if it already exists. Otherwise, it is generated automatically.

---

## BL\_EXECUTE\_CMD= Data Set Option

Specifies the operating system command for segment instances to run.

|              |                                                                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                       |
| Default:     | none                                                                                                                                           |
| Restriction: | Only for web tables                                                                                                                            |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                   |
| Data source: | Greenplum, HAWQ                                                                                                                                |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                                                                                                       |
| See:         | <a href="#">BL_EXECUTE_LOCATION= data set option</a> , <a href="#">BL_EXTERNAL_WEB= data set option</a> , <a href="#">BULKLOAD= set option</a> |

---

## Syntax

**BL\_EXECUTE\_CMD=***command | script*

### Syntax Description

#### ***command***

specifies the operating system command for segment instances to run.

#### ***script***

specifies a script that contains one or more operating system commands for segment instances to run.

## Details

Output is web table data at the time of access. Web tables that you access with an EXECUTE clause run the specified script or shell command on the specified hosts. By default, all active segment instances on all segment hosts run the command. For example, if each segment host runs four primary segment instances, the command is executed four times per host. You can also limit the number of segment instances that execute the command.

---

## BL\_EXECUTE\_LOCATION= Data Set Option

Specifies which segment instances runs the given command.

|              |                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                         |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                 |
| Default:     | none                                                                                                                                                                                             |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                     |
| Data source: | Greenplum, HAWQ                                                                                                                                                                                  |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                                                                                                                                                         |
| See:         | <a href="#">BL_EXECUTE_CMD= data set option</a> , <a href="#">BL_EXTERNAL_WEB= data set option</a> ,<br><a href="#">BL_LOCATION= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

```
BL_EXECUTE_LOCATION=ALL
BL_EXECUTE_LOCATION=PRIMARY
BL_EXECUTE_LOCATION=HOST [segment-host], number-of-segments
BL_EXECUTE_LOCATION=SEGMENT segmentID
```

### Syntax Description

#### ALL

specifies that all segment instances run the given command or script.

#### PRIMARY

specifies that the primary segment instance runs the given command or script.

#### HOST [segment-hostname], number-of-segments

indicates that the specified number of segments on the specified host runs the given command or script.

#### SEGMENT segmentID

indicates that the specified segment instance runs the given command or script.

---

## Details

For more information about valid values for this option, see the documentation for your database engine.

---

## BL\_EXTERNAL\_WEB= Data Set Option

Specifies whether the external data set accesses a dynamic data source.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: NO

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Greenplum, HAWQ

Note: Support for HAWQ was added in SAS 9.4M3.

See: [Accessing Dynamic Data in Web Tables](#), [BL\\_EXECUTE\\_CMD= data set option](#), [BL\\_EXECUTE\\_LOCATION= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_EXTERNAL\_WEB=YES | NO**

### Syntax Description

#### **YES**

specifies that the external data set is not a dynamic data source that resides on the web.

#### **NO**

specifies that the external data set is a dynamic data source that resides on the web.

---

## Details

The external data set can access a dynamic data source on the web, or it can run an operating system command or script. For more information about external web tables, see the documentation for your database engine.

## Example

```

libname sasfltl 'SAS-library';
libname mydblib greenplm user=myusr1 password=mypwd1 dsn=mysrv1;
proc sql;
create table mydblib.flights98
  (bulkload=yes
   bl_external_web='yes'
   bl_execute_cmd='/var/load_scripts/get_flight_data.sh'
   bl_execute_location='HOST'
   bl_format='TEXT'
   bl_delimiter='|')
  as select * from _NULL_;
quit;
libname sasfltl 'SAS-library';
libname mydblib greenplm user=myusr1 password=mypwd1 dsn=mysrv1;
proc sql;
create table mydblib.flights98
  (bulkload=yes
   bl_external_web='yes'
   bl_location_protocol='http'
   bl_datafile='intranet.company.com/expense/sales/file.csv'
   bl_format='CSV')
  as select * from _NULL_;
quit;

```

---

## BL\_FILE\_BADFILE= Data Set Option

Identifies a file that contains records that were rejected during bulk loading.

|              |                                                                                                                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                             |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                     |
| Aliases:     | BL_ERRORFILE<br>BL_BADFILE<br>BL_FILE_BADFILE<br>BL_FILE_LOG<br>BL_LOG                                                                                                                                                               |
| Default:     | none                                                                                                                                                                                                                                 |
| Restriction: | For use with a distributed system, this file name must point to a shared disk.                                                                                                                                                       |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                         |
| Data source: | SAP HANA                                                                                                                                                                                                                             |
| Note:        | Use this option to specify the fully qualified file name for the temporary error file.                                                                                                                                               |
| See:         | <a href="#">BL_CONTROL_FIELD_DELIMITER= data set option</a> ,<br><a href="#">BL_CONTROL_QUOTATION_MARK= data set option</a> ,<br><a href="#">BL_CONTROL_RECORD_DELIMITER= data set option</a> , <a href="#">BL_FILE_CONTROLFILE=</a> |

data set option, BL\_FILE\_DATAFILE= data set option, BL\_FILE\_DEFAULT\_DIR= data set option, BL\_FILE\_DELETE\_DATAFILE= data set option, BULKLOAD= data set option

---

## Syntax

**BL\_FILE\_BADFILE**=*default-dir/BL\_table\_name\_number.err*

---

## BL\_FILE\_CONTROLFILE= Data Set Option

Identifies the file that contains control statements.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Aliases: BL\_CONTROL

BL\_CONTROLFILE

BL\_FILE\_CONTROLFILE

Default: *default-dir/BL\_table\_name\_number.ctl*

Restriction: For use with a distributed system, this file name must point to a shared disk.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

Note: Use this option to specify the fully qualified file name for the temporary control file.

See: BL\_CONTROL\_FIELD\_DELIMITER= data set option,  
BL\_CONTROL\_QUOTATION\_MARK= data set option,  
BL\_CONTROL\_RECORD\_DELIMITER= data set option, BL\_FILE\_BADFILE= data set option, BL\_FILE\_DATAFILE= data set option, BL\_FILE\_DEFAULT\_DIR= data set option, BL\_FILE\_DELETE\_DATAFILE= data set option, BULKLOAD= data set option

---

## Syntax

**BL\_FILE\_CONTROLFILE**=*default-dir/BL\_number.ctl*

---

## Details

The SAP HANA interface creates the control file by using information from the input data and SAS/ACCESS options. The file contains Data Definition Language (DDL) definitions that specify the location of the data and how the data corresponds to the database table. It is used to specify exactly how the loader should interpret the data that you are loading from the DATA file (.dat file).

By default, SAS/ACCESS creates a control file in the temporary file directory that the UTILLOC= system option specifies or in another directory based on your default file specifications. If you do not specify this option and a control file does not already exist, a file is created in the temporary file directory (or in another directory that is based on your default file specifications). If you do not specify this option and a control file already exists, the SAP HANA interface reuses the file and replaces the contents with the new control statements.

## BL\_FILE\_DATAFILE= Data Set Option

Identifies the file that contains DBMS data for bulk loading.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Alias:       | BL_DATAFILE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Restriction: | For use with a distributed system, this file name must point to a shared disk.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Data source: | SAP HANA                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Note:        | Use this option to specify the fully qualified file name for the temporary data file.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| See:         | <a href="#">BL_CONTROL_FIELD_DELIMITER= data set option</a> ,<br><a href="#">BL_CONTROL_QUOTATION_MARK= data set option</a> ,<br><a href="#">BL_CONTROL_RECORD_DELIMITER= data set option</a> , <a href="#">BL_FILE_BADFILE= data set option</a> , <a href="#">BL_FILE_CONTROLFILE= data set option</a> , <a href="#">BL_DELETE_DATAFILE= data set option</a> , <a href="#">BL_FILE_DEFAULT_DIR= data set option</a> , <a href="#">BL_FILE_DELETE_DATAFILE= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_FILE\_DATAFILE=default-dir/BL\_table\_name\_number.dat**

## Details

By default, the SAP HANA interface creates a data file with a DAT file extension in the temporary file directory that is specified by the UTILLOC= system option (or in another directory that is based on your default file specifications). In addition, by default, the data file is deleted after the load is completed. To override this behavior, specify BL\_DELETE\_DATAFILE=NO.

---

## BL\_FILE\_DEFAULT\_DIR= Data Set Option

Specifies the default directory for the temporary files to use in bulk loading.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Aliases: BL\_DEFAULT\_DIR

BL\_FILE\_DEFAULT\_DIR

Default: '/tmp'

Requirements: To specify this option, you must first specify BULKLOAD=YES.

When you use a distributed system, the path must point to a shared disk.

Data source: SAP HANA

See: [BL\\_CONTROL\\_FIELD\\_DELIMITER= data set option](#),  
[BL\\_CONTROL\\_QUOTATION\\_MARK= data set option](#),  
[BL\\_CONTROL\\_RECORD\\_DELIMITER= data set option](#), [BL\\_FILE\\_BADFILE= data set option](#),  
[BL\\_FILE\\_CONTROLFILE= data set option](#), [BL\\_FILE\\_DATAFILE= data set option](#),  
[BL\\_FILE\\_DELETE\\_DATAFILE= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_FILE\_DEFAULT\_DIR='filepath'**

---

## BL\_FILE\_DELETE\_DATAFILE= Data Set Option

Specifies whether to delete only the data file or all files that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Aliases: BL\_DELETE\_DATAFILE

BL\_FILE\_DELETE\_DATAFILE

Default: YES

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

See: [BL\\_CONTROL\\_FIELD\\_DELIMITER= data set option](#),  
[BL\\_CONTROL\\_QUOTATION\\_MARK= data set option](#),  
[BL\\_CONTROL\\_RECORD\\_DELIMITER= data set option](#), [BL\\_FILE\\_BADFILE= data set option](#)

[option](#), [BL\\_FILE\\_CONTROLFILE=](#) data set option, [BL\\_FILE\\_DATAFILE=](#) data set option, [BL\\_FILE\\_DEFAULT\\_DIR=](#) data set option, [BULKLOAD=](#) data set option

## Syntax

**BL\_FILE\_DELETE\_DATAFILE=YES | NO**

### Syntax Description

#### YES

deletes the data file that the SAS/ACCESS engine creates for the DBMS bulk-load facility.

#### NO

does not delete the file.

## BL\_FILESYSTEM= Data Set Option

Specifies the name of the file system within the external storage system.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Access

Default: LIBNAME option value

Interaction: This option is used with the [BL\\_ACCOUNTNAME=](#), [BL\\_DNSSUFFIX=](#), and [BL\\_FOLDER=](#) options to specify the external data location.

Data source: Microsoft SQL Server

Note: Support for this data set option was added in SAS 9.4M7.

See: [BL\\_ACCOUNTNAME=](#) data set option, [BL\\_DNSSUFFIX=](#) data set option, [BL\\_FILESYSTEM=](#) LIBNAME option, [BL\\_FOLDER=](#) data set option

## Syntax

**BL\_FILESYSTEM="*file-system-name*"**

### Required Argument

#### *file-system-name*

specifies the name of the file system within the external storage system that files are to be read from and written to.

---

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see [“Bulk Loading to Azure Synapse Analytics” on page 1025](#).

The external data location is generated using this option and the BL\_ACCOUNTNAME=, BL\_DNSSUFFIX=, and BL\_FOLDER= data set options. These values are combined to define the URL to access an Azure Data Lake Storage Gen2 location:

`https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>`

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

---

## BL\_FOLDER= Data Set Option

Specifies the folder or file path for the data in the external storage system.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Access

Default: LIBNAME option value

Interaction: This option is used with the BL\_ACCOUNTNAME=, BL\_DNSSUFFIX=, and BL\_FILESYSTEM= options to specify the external data location.

Data source: Microsoft SQL Server

Note: Support for this data set option was added in SAS 9.4M7.

See: [BL\\_ACCOUNTNAME= data set option](#), [BL\\_DNSSUFFIX= data set option](#), [BL\\_FILESYSTEM= data set option](#), [BL\\_FOLDER= LIBNAME option](#)

---

## Syntax

**BL\_FOLDER="file-path-and-folder"**

### Required Argument

***file-path-and-folder***

specifies the folder or file path for the data in the external storage system. The location path begins with the container name.

## Details

This option is used when Microsoft SQL Server loads data to Azure Synapse Analytics (SQL DW) with an Azure Data Lake Storage Gen2 storage account. For more information, see “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

The external data location is generated using this option and the BL\_ACCOUNTNAME=, BL\_DNSSUFFIX=, and BL\_FILESYSTEM= data set options. These values are combined to define the URL to access an Azure Data Lake Storage Gen2 location:

```
https://<account-name>.<network-storage-host-name>/<file-system-name>/<file-path-folder>
```

These values might result in a URL similar to `https://myaccount.dfs.core.windows.net/myfilesystem/myfolder`.

---

## BL\_FORCE\_NOT\_NULL= Data Set Option

Specifies how to process CSV column values.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: NO

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Aster, Greenplum, HAWQ

Note: Support for HAWQ was added in SAS 9.4M3.

See: [BL\\_DELIMITER= data set option](#), [BL\\_FORMAT= data set option](#), [BL\\_NULL= data set option](#), [BL\\_QUOTE= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_FORCE\_NOT\_NULL=YES | NO**

### Syntax Description

#### YES

specifies that each specified column is processed as if it is enclosed in quotation marks and is therefore not a null value.

#### NO

specifies that each specified column is processed as if it is a null value.

---

## Details

You can use this option only when BL\_FORMAT=CSV. For the default null string, where no value exists between two delimiters, missing values are evaluated as zero-length strings.

---

## BL\_FORMAT= Data Set Option

Specifies the format of the external or web table data.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: TEXT

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Greenplum, Hadoop, HAWQ, PostgreSQL, Yellowbrick

Notes: Support for this data set option was added in SAS 9.4M1.

Support for HAWQ was added in SAS 9.4M3.

Support for Yellowbrick was added in SAS 9.4M7.

Support for Hadoop was added in SAS 9.4M8.

See: [BL\\_DELIMITER= data set option](#), [BL\\_FORCE\\_NOT\\_NULL= data set option](#), [BL\\_NULL= data set option](#), [BL\\_QUOTE= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_FORMAT=**[TEXT](#) | [CSV](#) | [ORC](#)

### Syntax Description

#### **TEXT**

specifies plain text format.

#### **CSV**

specifies a comma-separated value format.

**Restriction** CSV format is not supported for Hadoop.

#### **ORC**

specifies the Apache ORC (Optimized Row Columnar) open-source file format.

**Restriction** ORC format is supported only for Hadoop.

## Details

In Hadoop, this option controls the format of the bulk load staging file. The Hadoop engine does not support other bulk loading data set options for the staging file.

---

## BL\_FORMAT\_OPT= Data Set Option

Specifies format options for bulk loading CSV files.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: None

Restriction: Do not specify the COMPRESSION, ESCAPE, or FIELD\_DELIMITER options with BL\_FORMAT\_OPT=. Specify those options with the BL\_COMPRESS=, BL\_USE\_ESCAPE=, and BL\_DELIMITER= data set options, respectively.

Requirement: To specify this option, you must also specify BULKLOAD=YES.

Data source: Snowflake

Note: Support for this data set option was added in [SAS 9.4M8](#).

See: [BL\\_COMPRESS= data set option](#), [BL\\_DELIMITER= data set option](#), [BL\\_USE\\_ESCAPE= data set option on page 485](#), [BULKLOAD= data set option](#), [BULKLOAD= LIBNAME option](#)

Example:

```
data mylibref.mytable (bulkload=yes
                      bl_format_opt="FIELD_OPTIONALLY_ENCLOSED_BY='0X27' ");
  set sashelp.class;
run;
```

---

## Syntax

**BL\_FORMAT\_OPT="option" | "option=value"**

### Required Argument

**value**

specifies format options or option-value pairs for CSV files. Separate multiple options or option-value pairs with a space.

---

## BL\_HEADER= Data Set Option

Indicates whether to skip or load the first record in the input data file.

|              |                                                                                 |
|--------------|---------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)        |
| Categories:  | Bulk Loading<br>Data Set Control                                                |
| Default:     | NO                                                                              |
| Restriction: | You can use this option only when loading a table using an external web source. |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                    |
| Data source: | Greenplum, HAWQ                                                                 |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                                        |
| See:         | <a href="#">BULKLOAD= data set option</a>                                       |

## Syntax

**BL\_HEADER=YES | NO**

### Syntax Description

#### YES

indicates that the first record is skipped (not loaded).

#### NO

indicates that the first record is loaded.

## Details

When the first record of the input data file contains the name of the columns to load, you can indicate that it should be skipped during the load process.

## BL\_HOST= Data Set Option

Specifies the host name or IP address of the server where the external data file is stored.

|               |                                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                      |
| Categories:   | Bulk Loading<br>Data Set Control                                                                              |
| Default:      | DBMS-specific; SERVER= value, if SERVER= is set; otherwise, none [Impala]                                     |
| Requirements: | To specify this option, you must first specify BULKLOAD=YES.<br>You must enclose the name in quotation marks. |
| Data source:  | Aster, Greenplum, HAWQ, Impala                                                                                |
| Notes:        | Support for Impala was added in SAS 9.4M2.<br>Support for HAWQ was added in SAS 9.4M3.                        |

See: [BL\\_DBNAME= data set option](#), [BL\\_PATH= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_HOST='hostname'** [Aster, Impala]

**BL\_HOST='localhost'** [Greenplum, HAWQ]

## Syntax Description

### **localhost**

specifies the IP address of the server where the external data file is stored.

## Details

Use this option to pass the IP address to the DBMS bulk-load facility.

*Greenplum*: The default is 127.0.0.1. You can use the GPLOAD\_HOST environment variable to override the default.

## BL\_IAM\_ROLE= Data Set Option

Specifies an optional Amazon Web Services IAM role to use for the COPY command.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Access

Alias: BL\_ROLE=

Default: none

Data source: Amazon Redshift

Note: Support for this data set option was added in SAS 9.4M6.

See: [BL\\_KEY= data set option](#), [BL\\_SECRET= data set option](#), [BL\\_TOKEN= data set option](#)

Example: `BL_IAM_ROLE='1234567890:role/RedshiftRole'`

## Syntax

**BL\_IAM\_ROLE='role-value'**

## Syntax Description

### **role-value**

specifies the role value to use for the CREDENTIALS parameter in the AWS COPY command.

## Details

When BL\_IAM\_ROLE= is specified, the AWS COPY command uses the assigned value for the CREDENTIALS parameter. IAM roles are specified in the AWS environment using the following syntax. They are used as part of the authentication process.

```
aws_iam_role=arn:aws:iam:<AWS-account-ID>:role/<role-name>
```

For BL\_IAM\_ROLE=, use the following syntax:

```
BL_IAM_ROLE='<AWS-account-ID>:role/<role-name>'
```

SAS/ACCESS converts the value that you provide into the full format that is required for AWS.

For example, you might specify BL\_IAM\_ROLE='1234567890:role/RedshiftRole', where 1234567890 is the AWS account ID and role/RedshiftRole identifies the role resource. SAS/ACCESS converts this value to the full AWS IAM value arn:aws:iam::1234567890:role/RedshiftRole.

## BL\_IDENTITY= Data Set Option

Specifies the authentication method for the COPY command.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Access

Default: LIBNAME option value

Interaction: The value for this option and the BL\_SECRET= value are used for the CREDENTIAL argument with the COPY command with Azure Synapse Analytics (SQL DW).

Data source: Microsoft SQL Server

Note: Support for this data set option was added in SAS 9.4M7.

See: [BL\\_IDENTITY= LIBNAME option](#), [BL\\_SECRET= data set option](#)

Example: bl\_identity='Shared Access Signature'

## Syntax

**BL\_IDENTITY="identity-value"**

## Required Argument

### ***identity-value***

specifies the authentication method for the COPY command in Azure Synapse Analytics (SQL DW).

---

## Details

The BL\_IDENTITY= value is used as part of the CREDENTIAL argument that you specify for the COPY command in Azure Synapse Analytics (SQL DW). This functionality is supported when you are using an Azure Data Lake Gen2 account.

---

## BL\_IMPORT\_BATCH\_SIZE= Data Set Option

Specifies the number of records to insert in each commit.

|              |                                                                                                                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                               |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                       |
| Aliases:     | BL_BATCH_SIZE<br>BL_IMPORT_BATCH_SIZE                                                                                                                                                                  |
| Default:     | none                                                                                                                                                                                                   |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                           |
| Data source: | SAP HANA                                                                                                                                                                                               |
| See:         | <a href="#">BL_IMPORT_BATCH_SIZE=</a> , <a href="#">BL_IMPORT_OPTIONS=</a> , <a href="#">BL_IMPORT_TABLE_LOCK=</a> , <a href="#">BL_IMPORT_TYPE_CHECK=</a> , <a href="#">BULKLOAD=</a> data set option |

---

## Syntax

**BL\_IMPORT\_BATCH\_SIZE=batch-size**

---

## BL\_IMPORT\_OPTIONS= Data Set Option

Specifies additional options to add to the SAP HANA IMPORT statement.

|             |                                                                          |
|-------------|--------------------------------------------------------------------------|
| Valid in:   | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories: | Bulk Loading<br>Data Set Control                                         |
| Aliases:    | BCP_OPTIONS<br>BL_IMPORT_OPTIONS                                         |

BL\_MODIFIED\_BY

BL\_OPTIONS

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

Tip: Options are appended at the end of the generated IMPORT statement.

See: [BL\\_IMPORT\\_BATCH\\_SIZE=](#), [BL\\_IMPORT\\_TABLE\\_LOCK=](#),  
[BL\\_IMPORT\\_TYPE\\_CHECK=](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_IMPORT\_OPTIONS=options**

## BL\_IMPORT\_TABLE\_LOCK= Data Set Option

Specifies whether to lock the table for data import into column store tables.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Aliases: BL\_IMPORT\_TABLE\_LOCK

BL\_TABLE\_LOCK

Default: NO

Restriction: If you do not specify this option, then no locking occurs.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

See: [BL\\_IMPORT\\_BATCH\\_SIZE=](#), [BL\\_IMPORT\\_OPTIONS=](#), [BL\\_IMPORT\\_TYPE\\_CHECK=](#),  
[BULKLOAD= data set option](#)

## Syntax

**BL\_IMPORT\_TABLE\_LOCK=YES | NO**

## Syntax Description

### YES

specifies to add the TABLE LOCK option to the SAP HANA IMPORT statement.

### NO

specifies that no locking occurs.

---

## BL\_IMPORT\_TYPE\_CHECK= Data Set Option

Specifies whether to insert the record without checking the data type of each field.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Aliases: BL\_IMPORT\_TYPE\_CHECK

BL\_TYPE\_CHECK

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

See: [BL\\_IMPORT\\_BATCH\\_SIZE=](#), [BL\\_IMPORT\\_OPTIONS=](#), [BL\\_IMPORT\\_TABLE\\_LOCK=](#), [BULKLOAD=](#) data set option

---

## Syntax

**BL\_IMPORT\_TYPE\_CHECK=YES | NO**

### Syntax Description

#### **YES**

specifies to add the NO TYPE CHECK option to the SAP HANA IMPORT statement.

#### **NO**

specifies that no checking of data types occurs.

---

## BL\_INDEX\_OPTIONS= Data Set Option

Lets you specify SQL\*Loader Index options with bulk loading.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Alias: SQLDR\_INDEX\_OPTION=

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Oracle

See: [BULKLOAD=](#) data set option

## Syntax

**BL\_INDEX\_OPTIONS**=*any valid SQL\*Loader Index option segment-name*

### Syntax Description

#### ***any valid SQL\*Loader Index option***

The value that you specify for this option must be a valid SQL\*Loader index option, such as one of those below. Otherwise, an error occurs.

#### **SINGLEROW**

Use this option when loading either a direct path with APPEND on systems with limited memory or a small number of records into a large table. It inserts each index entry directly into the index, one record at a time. By default, DQL\*Loader does not use this option to append records to a table.

#### **SORTED INDEXES**

This clause applies when you are loading a direct path. It tells the SQL\*Loader that the incoming data has already been sorted on the specified indexes, allowing SQL\*Loader to optimize performance. It lets the SQL\*Loader optimize index creation by eliminating the sort phase for this data when using the direct-path load method.

---

## Details

You can now pass in SQL\*Loader index options when bulk loading. For details about these options, see the Oracle utilities documentation.

---

## Example: Specify SQL\*Loader Index Options

This example shows how you can use this option.

```
proc sql;
connect to oracle  ( user=myusr1 pw=mypwd1 path=mypath);
execute ( drop table blidxopts) by oracle;
execute ( create table blidxopts ( empno number, empname varchar2(20))) by oracle;
execute ( drop index blidxopts_idx) by oracle;
execute ( create index blidxopts_idx on blidxopts ( empno ) ) by oracle;
quit;
libname x oracle user=myusr1 pw=mypwd1 path=mypath;
data new;
empno=1; empname='one';
output;
empno=2; empname='two';
output;
run;
proc append base= x.blidxopts( bulkload=yes bl_index_options='sorted indexes
```

```
( blidxopts_idx)' ) data= new;  
run;
```

---

## BL\_INDEXING\_MODE= Data Set Option

Indicates which scheme the DB2 load utility should use for index maintenance.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | AUTOSELECT                                                               |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | DB2 under UNIX and PC Hosts                                              |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_INDEXING\_MODE=AUTOSELECT | REBUILD | INCREMENTAL | DEFERRED**

### Syntax Description

#### **AUTOSELECT**

The load utility automatically decides between REBUILD or INCREMENTAL mode.

#### **REBUILD**

All indexes are rebuilt.

#### **INCREMENTAL**

Indexes are extended with new data.

#### **DEFERRED**

The load utility does not attempt index creation if this mode is specified. Indexes are marked as needing a refresh.

---

## Details

For more information about using the values for this option, see the *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*.

---

## BL\_INTERNAL\_STAGE= Data Set Option

Specifies the internal stage and path.

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                      |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                  |
| Alias:       | BL_STAGE=                                                                                                                         |
| Default:     | none                                                                                                                              |
| Data source: | Snowflake                                                                                                                         |
| Note:        | Support for this data set option was added in the August 2019 release of SAS/ACCESS.                                              |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">BULKUNLOAD=data set option</a> |

## Syntax

**BL\_INTERNAL\_STAGE=stage-and-path**

## BL\_KEEPIDENTITY= Data Set Option

Determines whether the identity column that is created during bulk loading is populated with values that Microsoft SQL Server generates or with values that the user provides.

|              |                                                                                             |
|--------------|---------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                    |
| Categories:  | Bulk Loading<br>Data Set Control                                                            |
| Default:     | LIBNAME option value                                                                        |
| Restriction: | This option is valid only when you use the Microsoft SQL Server provider.                   |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                |
| Data source: | OLE DB                                                                                      |
| See:         | <a href="#">BL_KEEPIDENTITY= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_KEEPIDENTITY=YES | NO**

## Syntax Description

### YES

specifies that the user must provide values for the identity column.

### NO

specifies that the Microsoft SQL Server generates values for an identity column in the table.

---

## BL\_KEEPNULLS= Data Set Option

Indicates how NULL values in Microsoft SQL Server columns that accept NULL are handled during bulk loading.

|              |                                                                                                                      |
|--------------|----------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                             |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                     |
| Default:     | LIBNAME option value                                                                                                 |
| Restriction: | This option affects values in only Microsoft SQL Server columns that accept NULL and that have a DEFAULT constraint. |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                         |
| Data source: | OLE DB                                                                                                               |
| See:         | <a href="#">BL_KEEPNULLS= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a>                             |

---

## Syntax

**BL\_KEEPNULLS=YES | NO**

### Syntax Description

#### YES

preserves null values that the OLE DB interface inserts.

#### NO

replaces null values that the OLE DB interface inserts with a default value, as specified in the DEFAULT constraint.

---

## BL\_KEY= Data Set Option

Specifies the Amazon Web Services access key that is used with key-based access control. If you are using temporary token credentials, this is the temporary access key ID.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | none                                                                     |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | Amazon Redshift, Snowflake                                               |
| Notes:       | Support for this data set option was added in SAS 9.4M4.                 |

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

See: [BL\\_KEY= LIBNAME option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_KEY=key-value**

### Required Argument

**key-value**

specifies an AWS key that you use to access the AWS environment.

---

## Details

You can see more information about managing access keys in the [Amazon Web Services \(AWS\) documentation](#).

---

## BL\_LOAD\_METHOD= Data Set Option

Specifies the method by which data is loaded into an Oracle or PostgreSQL table during bulk loading.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: INSERT when loading an empty table; APPEND when loading a table that contains data

Restriction: REPLACE and TRUNCATE values apply only when you are loading data into a table that already contains data. In this case, you can use REPLACE and TRUNCATE to override the default value of APPEND. See your Oracle or PostgreSQL utilities documentation for information about using the TRUNCATE and REPLACE load methods.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Oracle, PostgreSQL

See: [BULKLOAD= data set option](#)

---

## Syntax

**BL\_LOAD\_METHOD=INSERT | APPEND | REPLACE | TRUNCATE**

**BL\_LOAD\_METHOD=APPEND | REPLACE | TRUNCATE**

## Syntax Description

### **INSERT**

requires the DBMS table to be empty before loading. (only Oracle)

### **APPEND**

appends rows to an existing DBMS table.

### **REPLACE**

deletes all rows in the existing DBMS table and loads new rows from the data file.

### **TRUNCATE**

uses the *SQL truncate* command to achieve the best possible performance. You must first disable the referential integrity constraints of the DBMS table.

## BL\_LOAD\_REPLACE= Data Set Option

Specifies whether DB2 appends or replaces rows during bulk loading.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: NO

Restriction: If BL\_LOAD\_REPLACE=YES and there is no data in the table that is replacing the DBMS table, then the DBMS table is unchanged.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under UNIX and PC Hosts

See: [BULKLOAD= data set option](#)

## Syntax

**BL\_LOAD\_REPLACE=YES | NO**

## Syntax Description

### **NO**

the CLI LOAD interface appends new rows of data to the DB2 table.

### **YES**

the CLI LOAD interface replaces the existing data in the table.

## BL\_LOCATION= Data Set Option

Specifies the location of a file on a web server for segment hosts to access.

|              |                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                    |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                            |
| Default:     | none                                                                                                                                        |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                |
| Data source: | Greenplum, HAWQ                                                                                                                             |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                                                                                                    |
| See:         | <a href="#">BL_EXECUTE_LOCATION= data set option</a> , <a href="#">BL_HOST= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_LOCATION=http://file-location**

## BL\_LOG= Data Set Option

Identifies a log file that contains information for bulk loading, such as statistics and errors.

|              |                                                                                                                                                                                                                              |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                     |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                             |
| Default:     | DBMS-specific                                                                                                                                                                                                                |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                 |
| Data source: | DB2 under UNIX and PC Hosts, Microsoft SQL Server, Oracle, Teradata                                                                                                                                                          |
| Note:        | Support for Microsoft SQL Server was added in SAS 9.4M7.                                                                                                                                                                     |
| See:         | <a href="#">BL_LOG= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">DB2 under UNIX and PC Hosts Bulk Loading</a> , <a href="#">Maximizing Teradata Load Performance</a> (Teradata bulk loading) |

## Syntax

**BL\_LOG=***path-and-log-file-name*

## Syntax Description

### *path-and-log-file-name*

specifies a file where information about the loading process is written.

## Details

See the reference section for your SAS/ACCESS interface for additional details about specifying this option.

When the DBMS bulk-load facility is invoked, it creates a log file. The contents of the log file are DBMS-specific. The BL\_ prefix distinguishes this log file from the one created by the SAS log. If BL\_LOG= is specified with the same path and file name as an existing log, the new log replaces the existing log.

*DB2 under UNIX and PC Hosts:* If BL\_LOG= is not specified, the log file is created in the temporary file directory that is specified by the UTILLOC= system option. For more information, see the bulk-load topic in the DB2 under UNIX and PC Hosts bulk-load section.

*Microsoft SQL Server:* The BL\_LOG= value is used with the COPY command in Azure Synapse Analytics (SQL DW). This functionality is supported when you are using an Azure Data Lake Gen2 account.

*Teradata:* For more information, see the bulk-load topic for Teradata.

*Oracle:* When the SQL\*Loader is invoked, it creates a log file. This file contains a detailed summary of the load, including a description of any errors. If SQL\*Loader cannot create a log file, execution of the bulk loading terminates. If a log file does not already exist, it is created in the temporary file directory that is specified by the UTILLOC= system option (or in another directory that is based on your default file specifications). If a log file does already exist, the SQL\*Loader reuses the file, replacing the contents with information from the new load. On most platforms, the default file name takes the form BL\_<table>\_<unique-ID>.log:

*table*

specifies the table name

*unique-ID*

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates the number.

---

## BL\_LOGFILE= Data Set Option

Identifies a log file that contains information for bulk loading, such as statistics and errors.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

Requirements: To specify this option, you must first specify BULKLOAD=YES.

You must specify both a path and a file name.

Data source: PostgreSQL, Yellowbrick

Note: Support for Yellowbrick was added in SAS 9.4M7.

See: [BULKLOAD= data set option, “Bulk Loading and Unloading for PostgreSQL”](#)

## Syntax

**BL\_LOGFILE=***path-and-log-file-name*

### Syntax Description

***path-and-log-file-name***

specifies a file where information about the loading process is written.

## BL\_MAPFILE= Data Set Option

Indicates whether to use a map file to load or export tables.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: NO

Requirement: To specify this option, you must first set BULKLOAD=YES or BULKUNLOAD=YES.

Data source: Aster

Note: Support for this data set option was added for SAS 9.4.

See: [BULKLOAD= LIBNAME option](#), [BULKLOAD= data set option](#), [BULKUNLOAD= LIBNAME option](#), [BULKUNLOAD= data set option](#)

## Syntax

**BL\_MAPFILE=**YES | NO

### Syntax Description

**YES**

specifies that a map file is used.

**NO**

specifies that a map file is not used.

## Details

You can use this option to pass connection and table information in a text (map) file to the loader or export application during bulk loading or unloading. The SAS/ACCESS engine automatically creates and deletes the map file for each bulk loading or unloading.

## Example: Passing Connection and Table Information

SAS uses the sasflt.flt98 map file in this example to provide the needed connection and table information for bulk loading.

```
LIBNAME sasflt 'SAS-data-library';
LIBNAME net_air ASTER user=myusr1 pwd=mypwd1
    server=air2 database=flights dimension=yes;

data net_air.flights98
    (bulkload=YES bl_mapfile=yes);
    set sasflt.flt98;
run;
```

---

## BL\_MAXERRORS= Data Set Option

Specifies the maximum number of rejected rows that are allowed in the load before the COPY command is canceled.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Bulk Loading                                                             |
| Default:     | LIBNAME option value                                                     |
| Data source: | Microsoft SQL Server                                                     |
| Note:        | Support for this data set option was added in SAS 9.4M7.                 |
| See:         |                                                                          |

---

## Syntax

**BL\_MAXERRORS=***value*

### Required Argument

***value***

specifies the maximum number of rejected rows that are allowed in the load before the COPY command is canceled.

---

## Details

This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account.

---

## BL\_METHOD= Data Set Option

Specifies the bulk-loading method to use for DB2.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: none

Requirement: You must specify BULKLOAD=YES to specify this option.

Data source: DB2 under UNIX and PC Hosts

See: [BULKLOAD= data set option](#)

---

## Syntax

**BL\_METHOD=CLILOAD | IMPORT | LOAD**

### Required Arguments

#### **CLILOAD**

causes SAS/ACCESS to use the DB2 command-line interface to call the LOAD utility.

#### **IMPORT**

causes SAS/ACCESS to use the IMPORT utility.

---

#### **Note:**

If you are using SAS on a PC or UNIX platform but your data resides on DB2 on a z/OS platform, then you can only specify IMPORT for this option.

---

#### **LOAD**

causes SAS/ACCESS to use the LOAD utility.

---

## BL\_NULL= Data Set Option

Specifies the string that represents a null value.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: 'N' (TEXT mode), unquoted empty value (CSV mode)

Requirement: To specify this option, you must first specify BULKLOAD=YES.

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | Greenplum, HAWQ, PostgreSQL, Yellowbrick                                                                                                                                                                                               |
| Notes:       | Support for this data set option was added in SAS 9.4M1.<br>Support for HAWQ was added in SAS 9.4M3.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                |
| See:         | <a href="#">BL_DELIMITER=</a> data set option, <a href="#">BL_FORCE_NOT_NULL=</a> data set option,<br><a href="#">BL_FORMAT=</a> data set option, <a href="#">BL_QUOTE=</a> data set option, <a href="#">BULKLOAD=</a> data set option |

## Syntax

**BL\_NULL='/N' | empty-value**

## Details

You might prefer an empty string even in TEXT mode for cases where you do not want to distinguish nulls from empty strings. When you use this option with external and web tables, any data item that matches this string is considered a null value.

## BL\_NUM\_DATAFILES= Data Set Option

Specifies the number of data files to create to contain the source data.

|              |                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                          |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                  |
| Alias:       | BL_NUM_DATA_FILES=                                                                                                                                                                                |
| Default:     | 2                                                                                                                                                                                                 |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                      |
| Data source: | Amazon Redshift, Microsoft SQL Server, Snowflake                                                                                                                                                  |
| Notes:       | Support for this data set option was added in SAS 9.4M4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7. |
| Tip:         | You can also use this option with the BULKUNLOAD= option.                                                                                                                                         |

## Syntax

**BL\_NUM\_DATAFILES=value**

## Required Argument

### **value**

specifies the number of files to create to hold your source data. This value can be an integer between 1 and 100, inclusively.

---

## Details

*Amazon Redshift, Snowflake:* You might increase performance if you specify a number that corresponds to a multiple of the number of slices in your cluster. For more information, see the [Amazon Web Services documentation](#).

*Microsoft SQL Server:* This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account.

---

## BL\_NUM\_READ\_THREADS= Data Set Option

Specifies the number of threads to use for bulk unloading from the DBMS into SAS.

|              |                                                                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                     |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                             |
| Defaults:    | LIBNAME option value [Amazon Redshift, Google BigQuery]<br>none [Snowflake]                                                                                                                  |
| Range:       | 1–10                                                                                                                                                                                         |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                 |
| Data source: | Amazon Redshift, Google BigQuery, Snowflake                                                                                                                                                  |
| Notes:       | Support for this data set option was added in SAS 9.4M6.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Google BigQuery was added in SAS 9.4M7. |
| See:         | <a href="#">BL_NUM_READ_THREADS= LIBNAME option</a> , <a href="#">BULKUNLOAD= data set option</a> ,<br><a href="#">BULKUNLOAD= LIBNAME option</a>                                            |

---

## Syntax

**BL\_NUM\_READ\_THREADS=value**

## Required Argument

### **value**

specifies the number of threads that are used to perform bulk unloading (retrieval) of data from a DBMS.

---

## BL\_OPTIONS= Data Set Option

Passes options to the DBMS bulk-load facility, which affects how it loads and processes data.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: DBMS-specific

Requirements: To specify this option, you must first specify BULKLOAD=YES.

You must separate multiple options with commas (except in DB2) and enclose the entire string of options in single quotation marks.

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Microsoft SQL Server, Netezza, OLE DB, Oracle, SAP IQ, Snowflake

Notes: Support for Amazon Redshift was added in SAS 9.4M4.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Microsoft SQL Server was added in SAS 9.4M7.

See: [BL\\_OPTIONS= LIBNAME option](#), [BULKLOAD= LIBNAME option](#), [BULKLOAD= data set option](#)

Example: For Amazon Redshift, this option adds the ROUNDEC option to the end of the COPY command that is passed to the S3 tool.

`BL_OPTIONS='ROUNDEC'`

---

## Syntax

Form 1: Amazon Redshift, DB2 under UNIX and PC Hosts, OLE DB, Oracle:

**`BL_OPTIONS='option-1<, option-2<, ... ,option-N>>'`**

Form 2: DB2 under UNIX and PC Hosts:

**`BL_OPTIONS='option-1 <option-2 < ... option-N>>'`**

Form 3: Aster, Netezza, SAP IQ:

**`BL_OPTIONS='option-1 value-1<,option-2 value-2<...option-N value-N>>'`**

## Syntax Description

### ***option-N***

specifies an option from the available options that are specific to each SAS/ACCESS interface.

### ***option-N value-N***

specifies an option and the value to assign to that option. The available options are specific to each SAS/ACCESS interface.

---

## Details

You can use BL\_OPTIONS= to pass options to the DBMS bulk-load facility when it is called, which affects how data is loaded and processed.

*Amazon Redshift*: By default, no options are specified. Any options that you specify are added to the end of the COPY command that is executed by the S3 bulk load tool.

*Aster*: By default, no options are specified.

*DB2 under UNIX and PC Hosts*: This option passes DB2 file-type modifiers to DB2 LOAD or IMPORT commands to affect how data is loaded and processed. Not all DB2 file type modifiers are appropriate for all situations. You can specify one or more DB2 file type modifiers with IXF files. For a list of file type modifiers, see the description of the LOAD and IMPORT utilities in the *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*.

*Microsoft SQL Server*: This option is used to support bulk loading with the COPY command when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account.

*Netezza*: Any text that you enter for this option is appended to the USING clause of the CREATE EXTERNAL TABLE statement—namely, any external\_table\_options in the *Netezza Database User's Guide*.

*OLE DB*: By default, no options are specified. This option is valid only when you are using the Microsoft SQL Server provider. This option takes the same values as the -h HINT option of the Microsoft BCP utility. For example, the ORDER= option specifies the sort order of data in the data file. Use it to improve performance if the file is sorted according to the clustered index on the table. See the Microsoft SQL Server documentation for a complete list of supported bulk copy options.

*Oracle*: This option lets you specify the SQL\*Loader options ERRORS= and LOAD=. The ERRORS= option specifies the number of insert errors that terminates the load. The default value of ERRORS=1000000 overrides the default value for the Oracle SQL\*Loader ERRORS= option, which is 50. LOAD= specifies the maximum number of logical records to load. If the LOAD= option is not specified, all rows are loaded. See your Oracle utilities documentation for a complete list of SQL\*Loader options that you can specify in BL\_OPTIONS=.

*SAP IQ*: By default, no options are specified. Any text that you enter for this option is appended to the LOAD TABLE command that the SAS/ACCESS interface uses for the bulk-loading process.

*Snowflake*: By default, no options are specified.

---

## Examples

### Example 1: Specify the Number of Permitted Errors

In this Oracle example BL\_OPTIONS= specifies the number of errors that are permitted during a load of 2,000 rows of data, where all listed options are enclosed in quotation marks.

```
bl_options='ERRORS=999,LOAD=2000'
```

## Example 2: Specify External Table Options

This Netezza example shows you how to use BL\_OPTIONS= to specify two different external table options, CTRLCHARS and LOGDIR:

```
data netlib.mdata(bulkload=yes bl_options="ctrlchars true logdir
'c:\temp'");
set saslib.transdata;
run;
```

## BL\_PARFILE= Data Set Option

Creates a file that contains the SQL\*Loader command-line options.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | none                                                                     |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | Oracle                                                                   |
| Tip:         | The parse file is deleted at the end of SQL*Loader processing.           |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

## Syntax

**BL\_PARFILE=<parse-file>**

### Syntax Description

#### *parse-file*

the name that you give the file that contains the SQL\*Loader command line options. It can also specify the path. If you do not specify a path, the file is created in the temporary file directory that is specified by the UTILLOC= system option.

## Details

This option prompts the SQL\*Loader to use the PARFILE= option. This SQL\*Loader option lets you specify SQL\*Loader command-line options in a file instead of as command-line options. Here is an example of how you can call the SQL\*Loader by specifying user ID and control options.

```
sqlldr userid=myusr1/mypwd1 control=example.ctl
```

You can also call it by using the PARFILE= option.

```
sqlldr parfile=example.par
```

Example.par now contains the USERID= and CONTROL= options. Security is a major advantage of using the BL\_PARFILE= option because the user ID and password are stored in a separate file.

Permissions on the file default to operating system defaults. Create the file in a protected directory to prevent unauthorized users from accessing its contents.

To display the contents of the parse file in the SAS log, use the SASTRACE=" , , d" option. The password is blocked out and replaced with xxxx, however.

---

## Example: Call the SQL\*Loader without and with BL\_PARFILE=

This example demonstrates how SQL\*Loader invocation differs when you specify the BL\_PARFILE= option.

```
libname x oracle user=myusr1 pw=mypwd1;

/* In this DATA step, call the SQL*Loader without BL_PARFILE=.
   sqlldr userid=myusr1/mypwd1@oraclev9
      control=bl_bltst_0.ctl log=bl_bltst_0.log
      bad=bl_bltst_0.bad discard=bl_bltst_0.dsc
*/
data x.bltst ( bulkload=yes );
c1=1;
run;

/* In this DATA step, call the SQL*Loader with BL_PARFILE=.
   sqlldr parfile=test.par

   In this case all options are written to the test.par file.
*/
data x.bltst2 ( bulkload=yes bl_parfile='test.par');
c1=1;
run;
```

---

## BL\_PATH= Data Set Option

Specifies the path to use for bulk loading.

|               |                                                                                                                      |
|---------------|----------------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                             |
| Categories:   | Bulk Loading<br>Data Set Control                                                                                     |
| Default:      | none                                                                                                                 |
| Requirements: | To specify this option, you must first specify BULKLOAD=YES.<br>You must enclose the entire path in quotation marks. |
| Data source:  | Aster                                                                                                                |

See: [BL\\_DBNAME= data set option](#), [BL\\_HOST= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_PATH='path'**

### Syntax Description

**path**

specifies the path to use for bulk loading.

## Details

Use this option to pass the path to the DBMS bulk-load facility.

## BL\_PORT= Data Set Option

Specifies the port number to use.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Aliases: BULKLOAD\_PORT= [Hadoop, Impala]

BLPORT= [Impala]

Defaults: 8020 [Hadoop]

8080 [Greenplum, HAWQ]

50070 [Impala]

Restriction: *Hadoop*: This option is required if Hadoop HDFS service is running on a port other than 8020.

Requirement: To specify this option, you must first specify BULKEXTRACT= YES or BULKLOAD=YES.

Interaction: *Greenplum*: Use the GPOLOAD\_PORT environment variable to specify the port for the bulk-loading utility. The BL\_PORT= data set option overrides the value of the GPOLOAD\_PORT environment variable.

Data source: Greenplum, Hadoop, HAWQ, Impala

Notes: Support for Impala was added in SAS 9.4M2.

Support for HAWQ was added in SAS 9.4M3.

See: [BULKLOAD= data set option](#)

Examples: Specify the GPOLOAD\_PORT value on PC hosts in the Advanced system settings:

```
GPOLOAD_PORT 8000
```

Export the GPLOAD\_PORT value on UNIX hosts for the C shell:

```
setenv GPLOAD_PORT=8000
```

Specify the GPLOAD\_PORT value within the SAS invocation command:

```
-set GPLOAD_PORT 8000
```

---

## Syntax

**BL\_PORT=***port*

### Syntax Description

*port*

specifies the port number to use.

---

## Details

Use this option to specify the port number that bulk loading uses to communicate with the server where the input data file resides.

---

## BL\_PORT\_MAX= Data Set Option

Specifies the highest available port number for concurrent uploads.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

Requirements: To specify this option, you must first specify BULKLOAD=YES.

To reserve a port range, you must specify values for this option and also the BL\_PORT\_MIN= option.

Data source: DB2 under UNIX and PC Hosts

See: [BL\\_PORT\\_MIN= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_PORT\_MAX=***integer*

## Syntax Description

***integer***

specifies a positive integer that represents the highest available port number for concurrent uploads.

## BL\_PORT\_MIN= Data Set Option

Specifies the lowest available port number for concurrent uploads.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

Requirements: To specify this option, you must first specify BULKLOAD=YES.

To reserve a port range, you must specify values for both this option and the BL\_PORT\_MAX= option.

Data source: DB2 under UNIX and PC Hosts

See: [BL\\_PORT\\_MAX= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_PORT\_MIN=*integer***

## Syntax Description

***integer***

specifies a positive integer that represents the lowest available port number for concurrent uploads.

## BL\_PRESERVE\_BLANKS= Data Set Option

Determines how the SQL\*Loader handles requests to insert blank spaces into CHAR/VARCHAR2 columns with the NOT NULL constraint.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: NO

Restriction: This option is not supported on z/OS.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Oracle  
 See: [BULKLOAD= data set option](#)

## Syntax

**BL\_PRESERVE\_BLANKS=YES | NO**

### Syntax Description

#### YES

specifies that blank values are inserted as blank spaces.

---

#### CAUTION

When this option is set to YES, *any trailing blank spaces are also inserted*. For this reason, use this option with caution. It is recommended that you set this option to YES only for CHAR columns. Do not set this option to YES for VARCHAR2 columns because trailing blank spaces are significant in VARCHAR2 columns.

---

#### NO

specifies that blank values are inserted as null values.

## BL\_PROTOCOL= Data Set Option

Specifies the protocol to use.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)  
 Categories: Bulk Loading  
              Data Set Control  
 Default: gpfdist  
 Requirement: To specify this option, you must first specify BULKLOAD=YES.  
 Data source: Greenplum, HAWQ  
 Note: Support for HAWQ was added in SAS 9.4M3.  
 See: [BL\\_DATAFILE= data set option](#), [BL\\_HOST= data set option](#), [BULKLOAD= data set option](#), [Protocols for Accessing External Tables](#), [Using the file:// Protocol](#)

## Syntax

**BL\_PROTOCOL=gpfdist | file | http**

## Syntax Description

***gpfdist***

specifies the Greenplum file distribution program.

***file***

specifies external tables on a segment host.

***http***

specifies web address of a file on a segment host. This value is valid only for external web tables.

## BL\_PSQL\_PATH= Data Set Option

Specifies the location of the PSQL executable file.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: psql

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: PostgreSQL

See: [BULKLOAD= data set option](#)

## Syntax

**BL\_PSQL\_PATH=*path-name***

## Syntax Description

***path-name***

specifies the full path name to the PSQL executable file to let the SAS/ACCESS interface call the PSQL utility.

## BL\_QUOTE= Data Set Option

Specifies the quotation character for CSV mode.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: " (double quotation mark)

Requirements: BL\_FORMAT=CSV

MULTILOAD=YES

Data source: Aster, Greenplum, HAWQ, PostgreSQL, Yellowbrick

Notes: Support for this data set option was added in SAS 9.4M1.

Support for HAWQ was added in SAS 9.4M3.

Support for Yellowbrick was added in SAS 9.4M7.

See: [BL\\_DELIMITER= data set option](#), [BL\\_ESCAPE= data set option](#),  
[BL\\_FORCE\\_NOT\\_NULL= data set option](#), [BL\\_FORMAT= data set option](#), [BL\\_NULL= data set option](#), [BULKLOAD= data set option](#)

## Syntax

**BL\_QUOTE='single 1-byte character'**

### Syntax Description

#### **single 1-byte character**

specifies the quoting character to be used when a data value is quoted. This must be a single one-byte character or an octal code that represents a 1-byte character.

## Details

This option is allowed only when BL\_FORMAT=CSV. The default is a double quotation mark (""). If your data contains the double quotation mark, then set BL\_QUOTE= to a character that is not included in the data.

You can specify octal codes to indicate the character to use as a quoting character, such as \001 (Ctrl-A). Specify the octal code within quotation marks, such as BL\_QUOTE='\001'.

The characters that are assigned to BL\_QUOTE= and to BL\_DELIMITER= must be different.

## BL\_RECOVERABLE= Data Set Option

Determines whether the LOAD process is recoverable.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: DBMS-specific

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under UNIX and PC Hosts, Oracle

See: [BULKLOAD= data set option](#)

## Syntax

**BL\_RECOVERABLE=YES | NO**

### Syntax Description

#### YES

specifies that the LOAD process is recoverable. For DB2, YES also specifies that [BL\\_COPY\\_LOCATION](#)= should specify the copy location for the data.

#### NO

specifies that the LOAD process is not recoverable.

---

## Details

*DB2 under UNIX and PC Hosts:* The default is NO.

*Oracle:* The default is YES. Set this option to NO to improve direct load performance. Specifying NO adds the UNRECOVERABLE keyword before the LOAD keyword in the control file.

---

#### CAUTION

Be aware that an unrecoverable load does not log loaded data into the redo log file. Therefore, media recovery is disabled for the loaded table. For more information about the implications of using the UNRECOVERABLE parameter in Oracle, see your Oracle utilities documentation.

---



---

## Example: Specify a Load as Unrecoverable

This example shows how to use BL\_RECOVERABLE= to specify that the load is unrecoverable.

```
data x.recover_no (bulkload=yes bl_recoverable=no); c1=1; run;
```

---

## BL\_REGION= Data Set Option

Specifies the Amazon Web Services (AWS) region from which data is being loaded.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: none

|              |                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                          |
| Data source: | Amazon Redshift, Snowflake                                                                                                            |
| Notes:       | Support for this data set option was added in SAS 9.4M4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS. |
| See:         | <a href="#">BL_REGION= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a>                                                 |

---

## Syntax

**BL\_REGION=***region*

### Required Argument

***region***

specifies the AWS region from which S3 data is being loaded. You must specify the region when using the S3 tool to load data.

---

## Details

For more information about AWS regions, see the [COPY Parameter Reference](#).

---

## BL\_REJECT\_LIMIT= Data Set Option

Specifies the reject limit count.

|              |                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                    |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                            |
| Default:     | none                                                                                                                                        |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES and then specify BL_REJECT_TYPE=.                                               |
| Interaction: | If the BL_EXCEPTION= data set option is not specified, then rejected records are dropped and are not saved to an exception table.           |
| Data source: | Greenplum, HAWQ                                                                                                                             |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                                                                                                    |
| See:         | <a href="#">BL_EXCEPTION= data set option</a> , <a href="#">BL_REJECT_TYPE= data set option</a> , <a href="#">BULKLOAD= data set option</a> |

## Syntax

**BL\_REJECT\_LIMIT=***number*

### Syntax Description

***number***

specifies the reject limit count either as a percentage (1 to 100) of total rows or as a number of rows.

## Details

When BL\_REJECT\_TYPE=PERCENT, the percentage of rows per segment is calculated based on the Greenplum database configuration parameter (gp\_reject\_percent\_threshold). The default value for this parameter Greenplum parameter is 300.

Input rows with format errors are discarded if the reject limit count is not reached on any Greenplum segment instance during the load operation.

Constraint errors result when violations occur to such constraints as NOT NULL, CHECK, or UNIQUE. A single constraint error causes the entire external table operation to fail. If the reject limit is not reached, rows without errors are processed and rows with errors are discarded.

---

## BL\_REJECT\_TYPE= Data Set Option

Indicates whether the reject limit count is a number of rows or a percentage of total rows.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Default: ROWS

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Interaction: If the BL\_EXCEPTION= data set option is not specified, then rejected records are dropped and are not saved to an exception table.

Data source: Greenplum, HAWQ

Note: Support for HAWQ was added in SAS 9.4M3.

See: [BL\\_REJECT\\_LIMIT= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_REJECT\_TYPE=**ROWS | PERCENT

## Syntax Description

### **ROWS**

specifies the reject limit count as a number of rows.

### **PERCENT**

specifies the reject limit count as a percentage of total rows. Valid values range from 1 to 100.

## BL\_REMOTE\_FILE= Data Set Option

Specifies the base file name and location of DB2 LOAD temporary files.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under UNIX and PC Hosts

See: [BL\\_SERVER\\_DATAFILE= data set option](#) (specifies the path from the server),  
[BULKLOAD= data set option](#), DB2 under z/OS Bulk Loading

## Syntax

**BL\_REMOTE\_FILE=***path-name-and-base-file-name*

## Syntax Description

### ***path-name-and-base-file-name***

the full path name and base file name to which DB2 appends extensions (such as .log, .msg, and .dat files) to create temporary files during load operations. By default, BL\_<table>\_<unique-ID> is the form of the base file name.

#### ***table***

specifies the table name.

#### ***unique-ID***

specifies a number that prevents collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates this number.

## Details

When you specify this option, the DB2 LOAD command is used instead of the IMPORT command. For more information about these commands, see the bulk-load topic in the DB2 under z/OS section.

For *path name*, specify a location on a DB2 server that is accessed exclusively by a single DB2 server instance, and for which the instance owner has Read and Write permissions. Make sure that each LOAD command is associated with a unique *path name-and-base-file-name* value.

---

## BL\_RETURN\_WARNINGS\_AS\_ERRORS= Data Set Option

Specifies whether SQL\*Loader (bulk-load) warnings should be displayed in SAS through the SYSERR macro as warnings or errors.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | NO                                                                       |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | Oracle                                                                   |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_RETURN\_WARNINGS\_AS\_ERRORS=YES | NO**

### Syntax Description

#### YES

specifies to return all SQLLDER warnings as errors, which SYSERR reflects.

#### NO

specifies to return all SQLLDER warnings as warnings.

---

## BL\_ROW\_DELIMITER= Data Set Option

Specifies an override character sequence that delimits rows or records in data during data transfer or retrieval during bulk loading.

|           |                                                                          |
|-----------|--------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
|-----------|--------------------------------------------------------------------------|

|              |                                                          |
|--------------|----------------------------------------------------------|
| Categories:  | Bulk Loading<br>Data Set Control                         |
| Default:     | \x0A\x02\x0A (line feed, Control-B, line feed)           |
| Data source: | SAP IQ                                                   |
| Note:        | Support for this data set option was added in SAS 9.4M5. |
| See:         | <a href="#">BL_COLUMN_DELIMITER=</a> data set option     |

## Syntax

**BL\_ROW\_DELIMITER='character-sequence'**

### Required Argument

#### *character-sequence*

specifies a sequence of one to four characters that delimit rows in your data. Choose a sequence that does not appear in your data. Specify this sequence as printable characters, hexadecimal values, or both. Specify hexadecimal values in the format '\xHH', where HH can represent any value from 00 to FF. For example, you might specify the three-character sequence `BL_ROW_DELIMITER='AB\x09'`, which represents the characters 'AB' followed by a tab character.

## BL\_SECRET= Data Set Option

Specifies the secret access key that is used to bulk load data.

|              |                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                          |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                  |
| Default:     | none                                                                                                                                                                                              |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                      |
| Interaction: | <i>Microsoft SQL Server:</i> The value for this option and the BL_IDENTITY= value are used for the CREDENTIAL argument with the COPY command in Azure Synapse Analytics (SQL DW).                 |
| Data source: | Amazon Redshift, Microsoft SQL Server, Snowflake                                                                                                                                                  |
| Notes:       | Support for this data set option was added in SAS 9.4M4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7. |
| See:         | <a href="#">BL_IDENTITY=</a> data set option, <a href="#">BL_KEY=</a> data set option, <a href="#">BL_SECRET= LIBNAME</a> option, <a href="#">BULKLOAD=</a> data set option                       |

## Syntax

**BL\_SECRET="secret-access-key"**

### Required Argument

#### **secret-access-key**

specifies the secret access key to access a data source.

**TIP** To increase security, you can encode the key value by using PROC PWENCODE.

## Details

When the secret access key is encoded with PROC PWENCODE, the value is decoded by SAS/ACCESS before passing the value to the data source.

*Amazon Redshift, Snowflake:* The *secret-access-key* value specifies the Amazon Web Services (AWS) secret access key or a temporary secret access key. An AWS secret access key is associated with the key ID that you specified with the BL\_KEY= data set option. If you are using temporary token credentials, this value is the temporary secret access key.

*Microsoft SQL Server:* This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account. The *secret-access-key* specifies the secret value for the CREDENTIAL argument of the COPY command.

If you use the Azure portal to get a Shared Access Signature, such as when you create a location for storing a bulk load log, use the resulting query string as the BL\_SECRET= value. In this case, specify BL\_IDENTITY='Shared Access Signature'.

---

## BL\_SERVER\_DATAFILE= Data Set Option

Specifies the name and location of the data file that the DBMS server instance sees.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Alias: BL\_DATAFILE=

Default: creates a data file in the temporary file directory that is specified by the UTILLOC= system option (or in another directory that is based on your default file specifications).

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: DB2 under UNIX and PC Hosts, SAP IQ

See: [BL\\_CLIENT\\_DATAFILE= data set option](#), [BL\\_DATAFILE= data set option](#),  
[BL\\_REMOTE\\_FILE= data set option](#), [BULKLOAD= data set option](#)  
["Bulk Loading for DB2 under UNIX and PC Hosts"](#), [Bulk Loading for SAP IQ](#)

## Syntax

**BL\_SERVER\_DATAFILE=***path-and-data-file-name*

### Syntax Description

***path-name-and-data-file-name***

specifies the fully qualified path name and file name of the data file to load, as seen by the DBMS server instance. By default, the base file name takes the form **BL\_<table>\_<unique-ID>**:

***table***

specifies the table name.

***unique-ID***

specifies a number that is used to prevent collisions in the event of two or more simultaneous bulk loadings of a particular table. The SAS/ACCESS engine generates the number.

## Details

**DB2 under UNIX and PC Hosts:** You must also specify a value for **BL\_REMOTE\_FILE=**. If the path to the data file from the DB2 server instance is different from the path to the data file from the client, you must use **BL\_SERVER\_DATAFILE=** to specify the path from the DB2 server. By enabling the DB2 server instance to directly access the data file that **BL\_DATAFILE=** specifies, this option facilitates use of the DB2 LOAD command. For more information about the LOAD command, see the bulk-load topic in the DB2 under z/OS section. To specify the path from the client, see the **BL\_DATAFILE=** data set option.

**SAP IQ:** To specify the path from the client, see the **BL\_CLIENT\_DATAFILE=** data set option, which is the client view of the data file.

## BL\_SFTP\_HOST= Data Set Option

Specifies the network name of the remote host with the OpenSSH secure shell daemon (SSHD) server running.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Aliases: **BL\_HOST**  
**BL\_SERVER**

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | BL_SFTP_HOST                                                                                                                                                                                                                           |
|              | BL_SFTP_SERVER                                                                                                                                                                                                                         |
| Default:     | SAP-HANA-server-name                                                                                                                                                                                                                   |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                           |
| Data source: | SAP HANA                                                                                                                                                                                                                               |
| See:         | BL_SFTP_OPTIONS= data set option, BL_SFTP_USER= data set option,<br>BL_SFTP_WAIT_MILLISECONDS= data set option, BULKLOAD= data set option,<br>FILENAME statement for the SFTP access method in <i>SAS Global Statements: Reference</i> |

---

## Syntax

**BL\_SFTP\_HOST='host'**

---

## Details

You can specify the name of the host or the IP address of the computer. In most cases, this should be the server name of the SAP HANA server. If you do not specify this option, the host name for SFTP defaults to the SAP HANA server name.

---

## BL\_SFTP\_OPTIONS= Data Set Option

Specifies additional configuration options for Secure File Transfer Protocol (SFTP) access, such as port numbers.

|              |                                                                                                                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                            |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                    |
| Alias:       | BL_SFTP_OPTIONS                                                                                                                                                                                                                     |
| Default:     | none                                                                                                                                                                                                                                |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                                                                        |
| Data source: | SAP HANA                                                                                                                                                                                                                            |
| See:         | BL_SFTP_HOST= data set option, BL_SFTP_USER= data set option,<br>BL_SFTP_WAIT_MILLISECONDS= data set option, BULKLOAD= data set option,<br>FILENAME statement for the SFTP access method in <i>SAS Global Statements: Reference</i> |

---

## Syntax

**BL\_SFTP\_OPTIONS=sftp-options**

---

## BL\_SFTP\_USER= Data Set Option

Specifies the user name.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Aliases: BL\_SFTP\_USER  
BL\_USER

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

See: [BL\\_SFTP\\_HOST= data set option](#), [BL\\_SFTP\\_OPTIONS= data set option](#),  
[BL\\_SFTP\\_WAIT\\_MILLISECONDS= data set option](#), [BULKLOAD= data set option](#),  
FILENAME statement for the SFTP access method in [SAS Global Statements: Reference](#)

---

## Syntax

**BL\_SFTP\_USER='user'**

---

## BL\_SFTP\_WAIT\_MILLISECONDS= Data Set Option

Specifies the Secure File Transfer Protocol (SFTP) response time in milliseconds.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Aliases: BL\_SFTP\_WAIT\_MILLISECONDS  
BL\_WAIT\_MILLISECONDS

Default: none

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: SAP HANA

See: [BL\\_SFTP\\_HOST= data set option](#), [BL\\_SFTP\\_OPTIONS= data set option](#),  
[BL\\_SFTP\\_USER= data set option](#), [BULKLOAD= data set option](#), FILENAME statement  
for the SFTP access method in [SAS Global Statements: Reference](#)

## Syntax

**BL\_SFTP\_MILLISECONDS=***option-string*

---

## BL\_SUPPRESS\_NULLIF= Data Set Option

Indicates whether to suppress the NULLIF clause for the specified columns to increase performance when a table is created.

|              |                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                        |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                |
| Default:     | NO                                                                                                                                              |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES. If you specify more than one column name, you must separate the names with spaces. |
| Data source: | Oracle                                                                                                                                          |
| See:         | <a href="#">BULKLOAD= data set option</a>                                                                                                       |

---

## Syntax

**BL\_SUPPRESS\_NULLIF=(***\_ALL\_=YES | NO*  
**BL\_SUPPRESS\_NULLIF=(***column-name-1*=YES | NO)*< ... column-name-N*=YES | NO*>*)

### Syntax Description

***column-name-N*=YES | NO**

specifies whether the NULLIF clause should be suppressed for the specified column in the table.

***\_ALL\_*=YES | NO**

specifies whether the NULLIF clause should be suppressed for all columns.

---

## Details

This option processes values from left to right. If you specify a column name twice or use the *\_ALL\_* value, the last value overrides the first value that you specified for the column.

---

**CAUTION**

If you set this option to YES and try to insert null values, unpredictable values are inserted into the column.

---

## Examples

### Example 1: Suppress NULLIF for Specific Table Columns

In this example, BL\_SUPPRESS\_NULLIF= in the DATA step suppresses the NULLIF clause for columns C1 and C5 in the table.

```
data x.suppressnullif2_yes (bulkload=yes BL_SUPPRESS_NULLIF=(c1=yes c5=yes)) ;
run;
```

### Example 2: Suppress NULLIF for All Table Columns

In this example, BL\_SUPPRESS\_NULLIF= in the DATA step suppresses the NULLIF clause for all columns in the table.

```
libname x oracle user=myusr1 pw=mypwd1 path=mypath;
%let num=1000000; /* 1 million rows */
data x.testlmn ( bulkload=yes
                  BL_SUPPRESS_NULLIF=( _all_ =yes )
                  rename=(year=yearx) );
set x.big1mil (obs= &num ) ;
run;
```

---

## BL\_TIMEOUT= Data Set Option

Specifies the maximum completion time for a storage task, in seconds.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Bulk Loading                                                             |
| Default:     | LIBNAME option value                                                     |
| Data source: | Microsoft SQL Server                                                     |
| Note:        | Support for this data set option was added in SAS 9.4M7.                 |
| See:         | <a href="#">BL_TIMEOUT= LIBNAME option</a>                               |

---

## Syntax

**BL\_TIMEOUT=***value*

## Required Argument

### **value**

specifies the maximum completion time for a storage task, expressed as a number of seconds. If the task does not complete within the specified time, the storage system might return an error.

---

## Details

This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account.

---

## BL\_TOKEN= Data Set Option

Specifies a temporary token that is associated with the temporary credentials that you specify with the BL\_KEY= and BL\_SECRET= data set options.

|              |                                                                                                                                                                             |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                    |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                            |
| Default:     | none                                                                                                                                                                        |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.                                                                                                                |
| Data source: | Amazon Redshift, Snowflake                                                                                                                                                  |
| Notes:       | Support for this data set option was added in SAS 9.4M4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.                                       |
| See:         | <a href="#">BL_KEY= data set option</a> , <a href="#">BL_SECRET= data set option</a> , <a href="#">BL_TOKEN= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> |

---

## Syntax

**BL\_TOKEN=***temporary-token*

## Required Argument

### ***temporary-token***

specifies a temporary token that is associated with the temporary credentials that you provide with BL\_KEY= and BL\_SECRET= data set options.

**Note:** You can safely ignore any notes you receive about your token being too long.

---

## Details

When you obtain the temporary credentials that you use to access the Amazon Web Services environment, you obtain an AWS key, an AWS secret key, and an AWS token. These credentials are valid only for a limited amount of time.

---

## BL\_USE\_MANIFEST= Data Set Option

Specifies whether to create a manifest file for S3. By default, SAS/ACCESS Interface to Amazon Redshift creates a manifest file that specifies all the data files that were used during a bulk-loading operation.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Categories:  | Bulk Loading<br>Data Set Control                                         |
| Default:     | YES                                                                      |
| Requirement: | To specify this option, you must first specify BULKLOAD=YES.             |
| Data source: | Amazon Redshift                                                          |
| Note:        | Support for this data set option was added in SAS 9.4M4.                 |
| See:         | <a href="#">BULKLOAD= data set option</a>                                |

---

## Syntax

**BL\_USE\_MANIFEST=YES | NO**

---

## BL\_USE\_ESCAPE= Data Set Option

Specifies whether to escape a predefined set of special characters during bulk loading or bulk retrieval.

|              |                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                          |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                  |
| Default:     | NO                                                                                                                                                                                                |
| Data source: | Amazon Redshift, Microsoft SQL Server, Snowflake                                                                                                                                                  |
| Notes:       | Support for this data set option was added in SAS 9.4M4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7. |
| See:         | <a href="#">BL_USE_ESCAPE= LIBNAME option</a> , <a href="#">BL_DELIMITER= data set option</a>                                                                                                     |

## Syntax

**BL\_USE\_ESCAPE=YES | NO**

### Required Arguments

#### YES

specifies that occurrences of a backslash ('\'), newline ('\n'), carriage return ('\r'), or user-defined delimiter receives a backslash inserted before it. The backslash preserves these occurrences in the data that is transferred during bulk loading or bulk retrieval. Any occurrence of the NULL character (0x00) is replaced by a space (' ').

#### NO

specifies that occurrences of a backslash ('\'), newline ('\n'), carriage return ('\r'), or user-defined delimiter are not escaped in the data that is transferred during bulk loading or bulk retrieval.

---

## Details

The predefined special characters that are treated as escape characters include the backslash ('\'), newline ('\n'), and return ('\r'). If you have specified another character for the BL\_DELIMITER= LIBNAME or data set option, that value is also treated as an escape character.

*Amazon Redshift:* A bulk-loading or bulk-unloading operation that is executed in Amazon Redshift uses the COPY command.

*Microsoft SQL Server:* This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account. Bulk loading to Azure Synapse Analytics uses the COPY command.

---

## BL\_USE\_LOG= Data Set Option

Specifies whether to get reports on rejected rows and the corresponding error file.

|              |                                                                                      |
|--------------|--------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)             |
| Category:    | Data Set Control                                                                     |
| Default:     | LIBNAME option value                                                                 |
| Data source: | Microsoft SQL Server                                                                 |
| Note:        | Support for this data set option was added in SAS 9.4M7.                             |
| See:         | <a href="#">BL_LOG= data set option</a> , <a href="#">BL_USE_LOG= LIBNAME option</a> |

## Syntax

**BL\_USE\_LOG=YES | NO**

### Required Arguments

#### YES

specifies that a log that contains rejected rows and corresponding errors should be created during bulk loading. The log is stored in the location that is specified by the BL\_LOG= option.

#### NO

specifies not to create a bulk loading log.

---

## Details

If no log location is specified for BL\_LOG=, then the bulk-loading log is stored in the temporary location that is specified by BL\_DEFAULT\_DIR=. Make sure that you have permission to write to the log location.

This option is used to support bulk loading when you access Azure Synapse Analytics (SQL DW) using an Azure Data Lake Gen2 account. To obtain Write permission for an Azure Synapse Analytics location, you might need to obtain a Shared Access Signature. For more information, see your Azure documentation.

---

## BL\_USE\_PIPE= Data Set Option

Specifies whether to use a named pipe for data transfer.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading

Data Set Control

Defaults: YES [Netezza, SAP IQ]

NO [Oracle, Greenplum, HAWQ]

Restrictions: Not available on z/OS [Oracle]

Applies to UNIX environments only [Greenplum, HAWQ]

Requirement: To specify this option, you must first specify BULKLOAD=YES [Greenplum, HAWQ, Netezza, Oracle, SAP IQ] or BULKUNLOAD=YES [Netezza, Oracle].

Data source: Greenplum, HAWQ, Netezza, Oracle, SAP IQ

Note: Support for Greenplum and HAWQ was added in SAS 9.4M3.

See: [BL\\_DATAFILE= data set option](#), [BULKLOAD= data set option](#), [BULKUNLOAD= LIBNAME option](#), [BULKUNLOAD= data set option](#)

## Syntax

**BL\_USE\_PIPE=YES | NO**

### Syntax Description

#### YES

specifies that a named pipe is used to transfer data between SAS/ACCESS interfaces and the DBMS client interface.

#### NO

specifies that a flat file is used to transfer data.

---

## Details

By default, the DBMS interface uses a named pipe interface to transfer large amounts of data between SAS and the DBMS when using bulk loading or unloading. If you prefer to use a flat data file that you can save for later use or examination, specify BL\_USE\_PIPE=NO.

---

### Example: Sample Code That Uses BL\_USE\_PIPE=YES

```
proc sql;
  create table mydblib.mileages
    (BULKLOAD=YES
     BL_USE_PIPE=YES
     BL_HOST='192.168.x.x'
     BL_PORT=8081)
    as select * from sashelp.mileages;
quit;
```

---

## BL\_USE\_SSL= Data Set Option

Specifies whether to use TLS encryption for connections to Amazon S3.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Bulk Loading  
Data Set Control

Defaults: none [Snowflake]  
YES [Amazon Redshift]

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Amazon Redshift, Snowflake

- Notes: Support for this data set option was added in SAS 9.4M4.  
Support for Snowflake was added in the August 2019 release of SAS/ACCESS.
- See: [BL\\_USE\\_SSL= LIBNAME option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_USE\_SSL=YES | NO**

---

## BL\_WARNING\_COUNT= Data Set Option

- Specifies the maximum number of row warnings to allow before the load fails.
- Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)
- Categories: Bulk Loading  
Data Set Control
- Default: 2147483646
- Requirement: To specify this option, you must first specify BULKLOAD=YES and also specify a value for BL\_REMOTE\_FILE=.
- Data source: DB2 under UNIX and PC Hosts
- See: [BL\\_REMOTE\\_FILE= data set option](#), [BULKLOAD= data set option](#)

---

## Syntax

**BL\_WARNING\_COUNT=***warning-count*

### Syntax Description

***warning-count***  
specifies the maximum number of row warnings to allow before the load fails.

---

## Details

Use this option to limit the maximum number of rows that generate warnings. See the log file for information about why the rows generated warnings.

---

## BL\_YB\_PATH= Data Set Option

Specifies the path to the Yellowbrick ybload and ybunload tools.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Bulk Loading

Default: none

Requirement: This option is required for bulk loading to or bulk unloading from a Yellowbrick database.

Data source: Yellowbrick

See: [BULKLOAD= data set option](#), [BULKUNLOAD= data set option](#), [YB\\_JAVA\\_HOME= data set option](#)

## Syntax

**BL\_YB\_PATH=<'>*path*<'>**

### Required Argument

***path***

specifies the path to the Yellowbrick ybload and ybunload tools. Enclose this value in single or double quotation marks if the path contains spaces or special characters.

## BUFFERS= Data Set Option Data Set Option

Specifies the number of shared memory buffers to use for transferring data from SAS to Teradata.

Valid in: DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)

Category: Data Set Control

Default: 2

Data source: Teradata

See: [MBUFSIZE= data set option](#), [MULTILOAD= data set option](#)

## Syntax

**BUFFERS=*number-of-shared-memory-buffers***

### Syntax Description

***number-of-shared-memory-buffers***

a numeric value between 1 and 8 that specifies the number of buffers used for transferring data from SAS to Teradata.

## Details

BUFFERS= specifies the number of data buffers to use for transferring data from SAS to Teradata. When you use the [MULTILOAD= data set option](#), data is transferred from SAS to Teradata using shared memory segments. The default shared memory buffer size is 64K. The default number of shared memory buffers used for the transfer is 2.

Use BUFFERS= to vary the number of buffers for data transfer from 1 to 8. Specify the MBUFSIZE= data set option to vary the size of the shared memory buffers from the size of each data row up to 1MB.

---

## BULK\_BUFFER= Data Set Option

Specifies the number of bulk rows that the SAS/ACCESS engine can buffer for output.

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                |
| Category:    | Data Set Control                                                                        |
| Default:     | 100                                                                                     |
| Data source: | SAP ASE                                                                                 |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">ENABLE_BULK= LIBNAME option</a> |

---

## Syntax

**BULK\_BUFFER=***numeric-value*

### Syntax Description

#### *numeric-value*

specifies the maximum number of rows that are allowed. This value depends on the amount of memory that is available to your system.

---

## Details

This option improves performance by specifying the number of rows that can be held in memory for efficient retrieval from the DBMS. A higher number signifies that more rows can be held in memory and accessed quickly during output operations.

---

## BULKLOAD= Data Set Option

Loads rows of data as one unit.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Aliases:     | BL_DB2LDUTIL= [DB2 under z/OS]<br>BCP= [ODBC, OLE DB, SAP IQ]<br>FASTLOAD= [Teradata]<br>YBLOAD= [Yellowbrick]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Defaults:    | YES [Hadoop, Spark]<br>NO [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata]<br>LIBNAME option value [Google BigQuery, JDBC]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Yellowbrick                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Notes:       | [Oracle] Important! The SQL*Loader that the <b>BULKLOAD=</b> data set option calls takes user name and password as command-line arguments. This means that a user who runs the <code>ps</code> command while the SQL*Loader is running could see the user credentials. To conceal the user name and password, be sure that the <b>BL_PARFILE=</b> data set option is also specified.<br><br>Support for PostgreSQL was added for SAS 9.4.<br>Support for Impala was added in SAS 9.4M2.<br>Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in SAS 9.4M4.<br>Support for JDBC was added in SAS Viya 3.4.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Microsoft SQL Server was added in SAS 9.4M7.<br>Support for Spark was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| Tips:        | You can also use this option with the <b>BULKUNLOAD=</b> option if your interface supports it.<br>[Impala] Before specifying <b>BULKLOAD=YES</b> , it is recommended that you set the <b>SAS_HADOOP_RESTFUL</b> environment variable to 1 before starting your SAS session. For more information, see <i>SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS</i> .<br>Using <b>BULKLOAD=YES</b> is the fastest way to insert rows into a DBMS table.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| See:         | <b>BULKUNLOAD=</b> LIBNAME option, <b>BULKUNLOAD=</b> data set option, <b>DBCOMMIT=</b> data set option, <b>ENABLE_BULK=</b> LIBNAME option [SAP ASE], <b>ERRLIMIT=</b> data set option, bulk loading details in the DBMS-specific reference section for your SAS/ACCESS interface                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

---

## Syntax

**BULKLOAD=YES | NO**

## Syntax Description

### **YES**

calls a DBMS-specific bulk-load facility to insert or append rows to a DBMS table.

### **NO**

uses the dynamic SAS/ACCESS engine to insert or append data to a DBMS table.

## Details

*Microsoft SQL Server:* The BULKLOAD= data set option enables Microsoft SQL Server to connect to Azure Synapse Analytics (SQL DW) with a Microsoft Azure Data Lake Storage Gen2 storage account. Bulk loading uses the COPY INTO command to load data into a DBMS table.

## BULKUNLOAD= Data Set Option

Rapidly retrieves (fetches) large number of rows from a data set.

|              |                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                                                                                                                                                                                                                                                                        |
| Alias:       | YBUNLOAD= [Yellowbrick]                                                                                                                                                                                                                                                                                                                                                                 |
| Defaults:    | NO [Netezza, PostgreSQL, Snowflake]<br>LIBNAME option value [Amazon Redshift, Aster, Google BigQuery]                                                                                                                                                                                                                                                                                   |
| Restriction: | To specify BULKUNLOAD=YES in Netezza, you must be the Netezza admin user, or you must have Create External Table permission.                                                                                                                                                                                                                                                            |
| Data source: | Amazon Redshift, Aster, Google BigQuery, Netezza, PostgreSQL, Snowflake, Yellowbrick                                                                                                                                                                                                                                                                                                    |
| Notes:       | Support for Amazon Redshift was added in SAS 9.4M6.<br>Support for Aster was added in SAS 9.4M2.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for PostgreSQL was added in SAS Viya 3.5.<br>Support for Google BigQuery was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7.                                            |
| Tip:         | Using BULKLOAD=YES is the fastest way to retrieve large numbers of rows from a DBMS table.                                                                                                                                                                                                                                                                                              |
| See:         | <a href="#">BULKUNLOAD= LIBNAME option</a> , <a href="#">BL_DATAFILE= data set option</a> , <a href="#">BL_DELIMITER= data set option</a> , <a href="#">BL_DELIMITER= data set option</a> , <a href="#">BL_USE_PIPE= data set option</a> , <a href="#">BULKLOAD= data set option</a> , “ <a href="#">Bulk Loading and Unloading for Aster</a> ” on page 742, bulk unloading for Netezza |

## Syntax

**BULKUNLOAD=YES | NO**

### Syntax Description

#### YES

enables bulk retrieval of data from the DBMS.

#### NO

uses standard result sets to retrieve data from the DBMS.

---

## Details

*Amazon Redshift*: This value calls the Amazon Redshift UNLOAD command to retrieve data from an Amazon Redshift DBMS.

*Aster*: This value uses the Aster client tool, ncluster\_export, to retrieve data from the Aster DBMS.

*Google BigQuery*: This value calls the Google BigQuery Extractor API.

*Netezza*: This value calls the Netezza Remote External Table interface to retrieve data from the Performance Server.

*PostgreSQL*: This value calls the PostgreSQL PSQL utility to retrieve data from a PostgreSQL table.

---

## BUSINESS\_DATATYPE= Data Set Option

specifies the data type of the BUS\_START and BUS\_END columns in tables that contain business time (temporal) data.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: TIMESTAMP(6)

Requirement: DB2 for z/OS

Data source: DB2 for z/OS

See: [TEMPORAL= data set option](#), [BUSINESS\\_TIMEFRAME= data set option](#), [SYSTEM\\_TIMEFRAME= data set option](#), [OVERLAPS= data set option](#)

---

## Syntax

**BUSINESS\_DATATYPE=DATE | TIMESTAMP(6)**

## Syntax Description

### **DATE**

specifies that the business data type is a date.

### **TIMESTAMP(6)**

specifies that the business data type is TIMESTAMP(6).

## **BUSINESS\_TIMEFRAME=** Data Set Option

lets you provide a date range or a datetime range to use when querying or modifying a temporal table.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: none

Requirement: DB2 z/OS with SAS 9.4

Data source: DB2 for z/OS

See: [TEMPORAL=](#) data set option, [SYSTEM\\_TIMEFRAME=](#) data set option, [OVERLAPS=](#) data set option, [BUSINESS\\_DATATYPE=](#) data set option

## Syntax

**BUSINESS\_TIMEFRAME=**FROM *date-or-datetime1* TO *date-or-datetime2*

## Syntax Description

### *date-or-datetimeN*

specifies the beginning or end of a time period that is used when you query or modify a table that contains temporal data. Use either a date or datetime value, depending on the value of the **BUSINES\_DATATYPE=** data set option.

## CAST= Data Set Option

Specifies whether SAS or the Teradata DBMS server should perform data conversions.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: NO

Restriction: CAST= applies only when you are reading Teradata tables into SAS, not when you are writing Teradata tables from SAS. It also applies only to SQL that SAS generates for you. If you provide your own SQL with the explicit SQL feature of PROC SQL, you must code your own casting clauses to force data conversions in Teradata instead of SAS.

|              |                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | Teradata                                                                                                                                                       |
| See:         | <a href="#">CAST= LIBNAME option</a> , <a href="#">CAST_OVERHEAD_MAXPERCENT= LIBNAME option</a> ,<br><a href="#">CAST_OVERHEAD_MAXPERCENT= data set option</a> |

## Syntax

**CAST=YES | NO**

### Syntax Description

#### YES

forces data conversions (casting) to be done on the Teradata DBMS server and overrides any data overhead percentage limit.

#### NO

forces data conversions to be done by SAS and overrides any data overhead percentage limit.

## Details

Internally, SAS numbers and dates are floating-point values. Teradata has several formats for numbers, including integers, floating-point values, and decimal values. Number conversion must occur when you are reading Teradata numbers that are not floating points (Teradata FLOAT). SAS/ACCESS can use the Teradata CAST= function to cause Teradata to perform numeric conversions. The parallelism of Teradata makes it suitable for performing this work, particularly if you are running SAS on z/OS, where CPU activity can be costly.

CAST= can cause more data to be transferred from Teradata to SAS, as a result of the option forcing the Teradata type into a larger SAS type. For example, the CAST= transfer of a Teradata BYTEINT to SAS floating point adds seven overhead bytes to each row transferred.

These Teradata types are candidates for casting:

- INTEGER
- BYTEINT
- SMALLINT
- DECIMAL
- DATE

SAS/ACCESS limits data expansion for CAST= to 20% to trade rapid data conversion by Teradata for extra data transmission. If casting does not exceed a 20% data increase, all candidate columns are cast. If the increase exceeds this limit, SAS attempts to cast Teradata DECIMAL types only. If casting only DECIMAL types still exceeds the increase limit, data conversions are done by SAS.

You can alter the casting rules by using either CAST= or  
CAST\_OVERHEAD\_MAXPERCENT= LIBNAME option. With

CAST\_OVERHEAD\_MAXPERCENT=, you can change the 20% overhead limit. With CAST=, you can override the percentage rules:

- CAST=YES forces Teradata to cast all candidate columns.
- CAST=NO cancels all Teradata casting.

---

## CAST\_OVERHEAD\_MAXPERCENT= Data Set Option

Specifies the overhead limit for data conversions to perform in Teradata instead of SAS.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: 20

Data source: Teradata

See: [CAST= LIBNAME option](#), [CAST= data set option](#), [CAST\\_OVERHEAD\\_MAXPERCENT= LIBNAME option](#)

---

## Syntax

**CAST\_OVERHEAD\_MAXPERCENT=< n >**

### Syntax Description

**< n >**

specifies any positive numeric value. The engine default is 20.

---

## Details

Teradata INTEGER, BYTEINT, SMALLINT, and DATE columns require conversion when read in to SAS. Either Teradata or SAS can perform conversions. When Teradata performs the conversion, the row size that is transmitted to SAS using the Teradata CAST operator can increase. CAST\_OVERHEAD\_MAXPERCENT= limits the allowable increase, also called *conversion overhead*.

For more information about conversions, conversion overhead, and casting, see the [CAST= LIBNAME option](#).

## Example: Increase the Allowable Overhead

This example demonstrates the use of `CAST_OVERHEAD_MAXPERCENT=` to increase the allowable overhead to 40%.

```
proc print data=mydblib.emp (cast_overhead_maxpercent=40);
  where empno<1000;
  run;
```

---

## CHAR\_AS\_BINARY= Data Set Option

Specifies whether to fetch all character columns as binary values when reading data from DB2.

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement                                                                                                      |
| Category:    | Data Set Control                                                                                                                  |
| Default:     | NO                                                                                                                                |
| Supports:    | NLS                                                                                                                               |
| Data source: | DB2 under UNIX and PC Hosts, DB2 under z/OS                                                                                       |
| Note:        | Support for this data set option was added in SAS 9.4M1.                                                                          |
| Tip:         | Setting this option to YES applies a \$HEXn format to all character variables. Data therefore is displayed in hexadecimal format. |

---

## Syntax

**CHAR\_AS\_BINARY=YES | NO**

---

## COLUMN\_DELIMITER= Data Set Option

Specifies the single character to use as a column (field) delimiter.

|              |                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                         |
| Category:    | Data Set Control                                                                                                                                 |
| Aliases:     | COL_DELIM=<br>COL_DELIMIT=<br>COL_DELIMITER=                                                                                                     |
| Default:     | \001 (Ctrl-A)                                                                                                                                    |
| Restriction: | You must specify a single character or a three-digit decimal value. Other commonly used formats (for example, '\t', 0x09, or '09'x) are invalid. |
| Requirement: | Specify a single-character or three-digit decimal ASCII value between 001 and 127.                                                               |

Data source: Hadoop  
 Note: Support for this data set option was added for SAS 9.4.  
 See: [ROW\\_DELIMITER= data set option](#)  
 Examples: COLUMN\_DELIMITER='#'  
 COLUMN\_DELIMITER=009 (tab)

## Syntax

**COLUMN\_DELIMITER=***single-character*

## COMMAND\_TIMEOUT= Data Set Option

Specifies the number of seconds to wait before a command times out.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)  
 Category: Data Set Control  
 Default: LIBNAME option value  
 Data source: OLE DB  
 See: [COMMAND\\_TIMEOUT= LIBNAME option](#)

## Syntax

**COMMAND\_TIMEOUT=***number-of-seconds*

## Syntax Description

### ***number-of-seconds***

an integer greater than or equal to 0, where 0 represents no time-out.

## CURSOR\_TYPE= Data Set Option

Specifies the cursor type for read only and updatable cursors.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)  
 Category: Data Set Control  
 Alias: CURSOR= [Impala, SAP IQ]  
 Default: LIBNAME option value  
 Data source: Amazon Redshift, DB2 under UNIX and PC Hosts, Impala, Microsoft SQL Server, ODBC, OLE DB, PostgreSQL, SAP IQ

- Note: Support for Amazon Redshift and PostgreSQL was added in SAS Viya 3.4.
- See: [COMMAND\\_TIMEOUT= LIBNAME option](#), [CURSOR\\_TYPE= LIBNAME option](#), [KEYSET\\_SIZE= data set option](#) [Microsoft SQL Server and ODBC]

---

## Syntax

**CURSOR\_TYPE=DYNAMIC | FORWARD\_ONLY | KEYSET\_DRIVEN | STATIC**

### Syntax Description

#### DYNAMIC

specifies that the cursor reflects all changes that are made to the rows in a result set as you move the cursor. The data values and the membership of rows in the cursor can change dynamically on each fetch.

---

**Note:** This value is not valid for Impala.

---

#### FORWARD\_ONLY

specifies that the cursor functions like a DYNAMIC cursor except that it supports only sequential fetching of rows.

---

**Note:** This value is not valid for OLE DB.

---

#### KEYSET\_DRIVEN

specifies that the cursor determines which rows belong to the result set when the cursor is opened. However, changes that are made to these rows are reflected as you move the cursor.

---

**Note:** This value is not valid for Impala.

---

#### STATIC

specifies that the cursor builds the complete result set when the cursor is opened. No changes made to the rows in the result set after the cursor is opened are reflected in the cursor. Static cursors are read-only.

---

## Details

Not all drivers support all cursor types. An error is returned if the specified cursor type is not supported. The driver is allowed to modify the default without an error. See your database documentation for more information.

When no options have been specified yet, here are the initial DBMS-specific default values.

**Table 13.2** DBMS-Specific Defaults for CURSOR\_TYPE=

| DB2 for UNIX and PC | Microsoft SQL Server | ODBC         | OLE DB       | Amazon Redshift<br>PostgreSQL<br>SAP IQ |
|---------------------|----------------------|--------------|--------------|-----------------------------------------|
| KEYSET_DRIVEN       | DYNAMIC              | FORWARD_ONLY | FORWARD_ONLY | DYNAMIC                                 |

Here are the operation-specific defaults.

**Table 13.3** Operation-Specific Defaults for CURSOR\_TYPE=

| Operation                              | DB2 for UNIX and PC | Microsoft SQL Server | ODBC, SAP IQ  | OLE DB                           |
|----------------------------------------|---------------------|----------------------|---------------|----------------------------------|
| insert<br>(UPDATE_SQL=NO)              | KEYSET_DRIVEN       | DYNAMIC              | KEYSET_DRIVEN | FORWARD_ONLY                     |
| read<br>(such as PROC PRINT)           | driver default      |                      |               | driver default<br>(FORWARD_ONLY) |
| update<br>(UPDATE_SQL=NO)              | KEYSET_DRIVEN       | DYNAMIC              | KEYSET_DRIVEN | FORWARD_ONLY                     |
| CONNECTION=GLOBAL<br>CONNECTION=SHARED |                     | DYNAMIC              |               | DYNAMIC                          |

**OLE DB:** Here are the OLE DB properties that are applied to an open rowset. For details, see your OLE DB programmer reference documentation.

**Table 13.4** OLE DB Properties for Rowsets

| CURSOR_TYPE=                               | OLE DB Properties Applied                                   |
|--------------------------------------------|-------------------------------------------------------------|
| FORWARD_ONLY or DYNAMIC<br>(see “Details”) | DBPROP_OTHERINSERT=TRUE,<br>DBPROP_OTHERUPDATEDELETE=TRUE   |
| KEYSET_DRIVEN                              | DBPROP_OTHERINSERT=FALSE,<br>DBPROP_OTHERUPDATEDELETE=TRUE  |
| STATIC                                     | DBPROP_OTHERINSERT=FALSE,<br>DBPROP_OTHERUPDATEDELETE=FALSE |

---

## DATETIME2= Data Set Option

Specifies the scale for the timestamp literal for Microsoft SQL Server 2008 and the native Microsoft driver.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: NO

Data source: Microsoft SQL Server, ODBC

Note: Support for Microsoft SQL Server was added in SAS Viya 3.4.

See: [DATETIME2= LIBNAME option](#)

---

## Syntax

**DATETIME2=YES | NO**

### Syntax Description

#### **YES**

specifies a DATETIME precision and scale when creating the timestamp for the WHERE clause.

#### **NO**

uses the first occurring DATETIME precision and scale when creating the timestamp for the WHERE.

---

## DB\_ONE\_CONNECT\_PER\_THREAD= Data Set Option

Specifies whether to limit the number of connections to the DBMS server for a threaded Read.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: YES

Data source: Oracle

See: [Autopartitioning Scheme for Oracle](#)

---

## Syntax

**DB\_ONE\_CONNECT\_PER\_THREAD=YES | NO**

### Syntax Description

#### **YES**

enables this option, allowing only one connection per partition.

#### **NO**

disables this option.

---

## Details

Use this option if you want to have only one connection per partition. By default, the number of connections is limited to the maximum number of allowed threads. If the value of the maximum number of allowed threads is less than the number of partitions on the table, a single connection reads multiple partitions.

---

## DBCOMMIT= Data Set Option

Causes an automatic COMMIT (a permanent writing of data to the DBMS) after a specified number of rows are processed.

|              |                                                                                                                                                                                                                                                                                                                                    |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                           |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                   |
| Alias:       | CHECKPOINT= [Teradata]                                                                                                                                                                                                                                                                                                             |
| Default:     | LIBNAME option value                                                                                                                                                                                                                                                                                                               |
| Interaction: | This option is not honored when the DBIDIRECTEXEC system option is enabled.                                                                                                                                                                                                                                                        |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                                               |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Impala was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                        |
| See:         | BULKLOAD= LIBNAME option, BULKLOAD= data set option, CONNECTION= LIBNAME option, DBCOMMIT= LIBNAME option, ERRLIMIT= LIBNAME option, ERRLIMIT= data set option, INSERT_SQL= LIBNAME option, INSERT_SQL= data set option, INSERTBUFF= LIBNAME option, INSERTBUFF= data set option, ML_CHECKPOINT= data set option, "Using FastLoad" |

## Syntax

**DBCOMMIT=n**

### Syntax Description

*n*

specifies an integer greater than or equal to 0.

---

## Details

DBCOMMIT= affects update, delete, and insert processing. The number of rows processed includes rows that are not processed successfully. When DBCOMMIT=0, COMMIT is issued only once—after the procedure or DATA step completes.

If you explicitly specify the DBCOMMIT= option, SAS/ACCESS fails any update with a WHERE clause.

If you specify both DBCOMMIT= and ERRLIMIT= and these options collide during processing, COMMIT is issued first and ROLLBACK is issued second. Because COMMIT is issued (through the DBCOMMIT= option) before ROLLBACK (through the ERRLIMIT= option), DBCOMMIT= overrides ERRLIMIT=.

*DB2 under UNIX and PC Hosts:* When BULKLOAD=YES, the default is 10000.

*Teradata:* For the default behavior of this option, see the FastLoad description in the Teradata section. DBCOMMIT= is disabled for MultiLoad to prevent any conflict with ML\_CHECKPOINT=.

*Vertica:* For updates or deletions, any value for DBCOMMIT= is reset to 0, because SAS/ACCESS can commit changes only once at the end of an action. Also, for updates or deletions, if a value is not already specified for the CONNECTION= LIBNAME option, then CONNECTION= is set to UNIQUE.

---

## Example: Specify the Number of Row to Process

A commit is issued after every 10 rows are processed in this example:

```
data oracle.dept (dbcommit=10);
      set myorlib.staff;
run;
```

---

## DBCONDITION= Data Set Option

Specifies criteria for subsetting and ordering DBMS data.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

|               |                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:      | none                                                                                                                                                                                                                                                                                                                                                 |
| Restrictions: | The DBKEY= and DBINDEX= options are ignored when you use DBCONDITION=. <i>Hadoop, JDBC</i> : DBCONDITION= is ignored if it specifies ORDER BY and you also use a BY statement.                                                                                                                                                                       |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                  |
| Notes:        | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:          | <a href="#">DBINDEX= data set option</a> , <a href="#">DBKEY= data set option</a>                                                                                                                                                                                                                                                                    |

## Syntax

**DBCONDITION="DBMS-SQL-query-clause"**

### Syntax Description

**DBMS-SQL-query-clause**

specifies a DBMS-specific SQL query clause, such as WHERE, GROUP BY, HAVING, or ORDER BY.

## Details

When you use this option to specify selection criteria in the form of DBMS-specific SQL query clauses, performance is often enhanced because the SAS/ACCESS engine passes these clauses directly to the DBMS for processing. The DBMS checks the criteria for syntax errors when it receives the SQL query.

## Example: Return Only Condition-Specific Rows

In this example, the function that is passed to the DBMS with the DBCONDITION= option causes the DBMS to return to SAS only those rows that satisfy the condition.

```
proc sql;
  create view smithnames as
    select lastname from myorlib.employees
```

```
(dbcondition="where soundex(lastname) = soundex('SMYTHE') " )
  using libname myoralib oracle user=myusr1
    pw=mypwd1 path=mysrv1;
select lastname from smithnames;
```

## DBCONSTRAINT= Data Set Option

Provides table-level definitions to specify when a table is created.

|              |                                                                             |
|--------------|-----------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)    |
| Category:    | Data Set Control                                                            |
| Default:     | none                                                                        |
| Requirement: | You must specify constraints within the CREATE TABLE statement parentheses. |
| Data source: | Teradata                                                                    |
| See:         | <a href="#">DBCREATE_TABLE_OPTS= data set option</a>                        |

## Syntax

**DBCONSTRAINT='DBMS-SQL-clauses'**

### Syntax Description

#### **DBMS-SQL-clauses**

indicates one or more clauses that are specific to Teradata that must be specified when creating a table but that must appear inside the CREATE TABLE parentheses.

## Details

Use this option to add table-level definitions in the CREATE TABLE statement. DBCREATE\_TABLE\_OPTS= is similar to this option except that it lets you add DBMS-specific text outside (to the right) of the parentheses.

## Example: Specify Primary Key Columns for a Table

In this example, DBCONSTRAINT= specifies a table-level constraint that columns x and y are primary key columns.

```
libname x teradata user=myusr1 pw=mypwd1;
```

```

/*
 * Submits this SQL with table-level constraints.
 *
 * CREATE MULTISET TABLE "test"
 * ("x" FLOAT NOT NULL ,
 *  "y" FLOAT NOT NULL ,
 *  CONSTRAINT test PRIMARY KEY(X,Y)
 * );
 */
data x.test(DBCONSTRAINT='CONSTRAINT test PRIMARY KEY(X,Y)' DBNULL=(_ALL_=NO));
x=1;y=1;
run;

```

## DBCREATE\_TABLE\_EXTERNAL= Data Set Option

Specifies the type of table to create and how associated data files are handled.

|              |                                                                                                                                                                                                                                 |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                        |
| Category:    | Data Set Control                                                                                                                                                                                                                |
| Aliases:     | DBCREATE_EXTERNAL=<br>DBCREATE_EXT=                                                                                                                                                                                             |
| Default:     | NO                                                                                                                                                                                                                              |
| Interaction: | You can specify this option, the DBCREATE_TABLE_LOCATION= option, or both.                                                                                                                                                      |
| Data source: | Hadoop                                                                                                                                                                                                                          |
| Tip:         | This option determines only the disposition of a file upon delete.                                                                                                                                                              |
| See:         | <a href="#">DBCREATE_TABLE_EXTERNAL= LIBNAME option</a> , <a href="#">DBCREATE_TABLE_LOCATION= data set option</a> , <a href="#">DBCREATE_TABLE_OPTS= LIBNAME option</a> , <a href="#">DBCREATE_TABLE_OPTS= data set option</a> |

## Syntax

**DBCREATE\_TABLE\_EXTERNAL=YES | NO**

### Syntax Description

#### **YES**

creates an *external* table—one that is stored outside of the Hive warehouse.

#### **NO**

creates a *managed* table—one that is managed within the Hive warehouse.

## Details

When a managed table is dropped, its data is also deleted. When an external table is dropped, its data is preserved. Create an EXTERNAL table if you want to preserve table data if the table is dropped. SAS issues a DROP TABLE statement when PROC DELETE references a Hive table and also with the DROP TABLE statement in PROC SQL.

## Example: Protect Data from DROP TABLE

In this example, DBCREATE\_TABLE\_LOCATION= stores the table data outside of the Hive warehouse. DBCREATE\_TABLE\_EXTERNAL=YES protects the data from being deleted if the table is dropped.

```
LIBNAME db HADOOP SERVER=mysrv1 USER=myusr1 DB=myschema1;
DATA db.mytab (
    DBCREATE_TABLE_EXTERNAL=YES
    DBCREATE_TABLE_LOCATION="/mydir/mytab");
SET mydata;
RUN;
```

---

## DBCREATE\_TABLE\_LOCATION= Data Set Option

Identifies the HDFS location of the root directory for storing table data.

|              |                                                                                                                                                                                                                                                                                           |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                  |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                          |
| Aliases:     | DBCREATE_LOCATION=<br>DBCREATE_LOC=<br>DBCREATE_PATH=                                                                                                                                                                                                                                     |
| Defaults:    | /user/hive/warehouse/ <i>tabname</i> (with the default schema)<br>/user/hive/warehouse/ <i>schema.db</i> / <i>tabname</i> (with a nondefault schema)                                                                                                                                      |
| Interaction: | You can specify this option, the DBCREATE_TABLE_EXTERNAL= option, or both.                                                                                                                                                                                                                |
| Data source: | Hadoop                                                                                                                                                                                                                                                                                    |
| Tip:         | This option determines only the physical location of a file.                                                                                                                                                                                                                              |
| See:         | <a href="#">DBCREATE_TABLE_LOCATION= LIBNAME option</a> , <a href="#">DBCREATE_TABLE_EXTERNAL= LIBNAME option</a> , <a href="#">DBCREATE_TABLE_EXTERNAL= data set option</a> , <a href="#">DBCREATE_TABLE_OPTS= LIBNAME option</a> , <a href="#">DBCREATE_TABLE_OPTS= data set option</a> |

---

## Syntax

**DBCREATE\_TABLE\_LOCATION='path'**

## Syntax Description

### 'path'

specifies the HDFS location of the root directory for storing table data.

---

## Details

Use this option to specify an alternative HDFS location, which adds the LOCATION keyword to the CREATE TABLE DDL.

---

## Example: Create a File in an Alternative Hive Depository

Both DBCREATE\_TABLE\_EXTERNAL= and DBCREATE\_TABLE\_LOCATION= data set options are specified in this example.

```
data db.mytab (
    dbccreate_table=external=yes
    dbccreate_table_location="/mydir/mytab");
set mydata;
run;
```

---

## DBCREATE\_TABLE\_OPTS= Data Set Option

Specifies DBMS-specific syntax to add to the end of the CREATE TABLE statement.

|              |                                                                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                      |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                              |
| Alias:       | POST_STMT_OPTS=                                                                                                                                                                                                                                                                                                               |
| Default:     | LIBNAME option value                                                                                                                                                                                                                                                                                                          |
| Restriction: | <i>Hadoop</i> : The value that you specify for this option cannot contain a PARTITIONED BY clause.                                                                                                                                                                                                                            |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                           |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for JDBC was added in SAS Viya 3.4.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7. |

- Tip: If you want all output tables to be in the default (non-TEXTFILE) format, use the LIBNAME option. (See the LIBNAME option for examples.)
- See: DBCREATE\_TABLE\_EXTERNAL= LIBNAME option, DBCREATE\_TABLE\_EXTERNAL= data set option, DBCREATE\_TABLE\_LOCATION= data set option, DBCREATE\_TABLE\_OPTS= LIBNAME option, DBTYPE= data set option, POST\_STMT\_OPTS= data set option, POST\_TABLE\_OPTS= data set option, PRE\_STMT\_OPTS= data set option, PRE\_TABLE\_OPTS= data set option

## Syntax

**DBCREATE\_TABLE\_OPTS='DBMS-SQL-clauses'**

### Required Argument

#### **DBMS-SQL-clauses**

specifies one or more DBMS-specific clauses that can be appended to the end of an SQL CREATE TABLE statement.

## Details

You can use this option to add DBMS-specific clauses at the end of the SQL CREATE TABLE statement. The SAS/ACCESS engine passes the SQL CREATE TABLE statement and its clauses to the DBMS. The DBMS then executes the statement and creates the DBMS table. This option applies only when you are creating a DBMS table by specifying a libref associated with DBMS data.

If you need to add an option in a location other than at the end of your CREATE TABLE statement, use one of these data set options: POST\_TABLE\_OPTS=, PRE\_STMT\_OPTS=, and PRE\_TABLE\_OPTS=. For example, for Greenplum, a WITH clause should appear after the table name but before a DISTRIBUTED RANDOMLY clause in a CREATE TABLE statement. You should therefore specify a WITH clause using the POST\_TABLE\_OPTS= data set option.

## Examples

### Example 1: Partition a DB2 Table

In this example, the DB2 table TEMP is created with the value of the DBCREATE\_TABLE\_OPTS= option appended to the CREATE TABLE statement.

```
libname mydblib db2 user=myusr1 pwd=mypwd1 dsn=sample;
data mydblib.temp (DBCREATE_TABLE_OPTS='PARTITIONING KEY (X) USING HASHING');
x=1; output;
x=2; output;
run;
```

When you use this data set option to create the DB2 table, the SAS/ACCESS Interface to DB2 passes this DB2 SQL statement:

```
CREATE TABLE TEMP (X DOUBLE) PARTITIONING KEY (X) USING HASHING
```

## Example 2: Partition a Hive Table

In this example, a Hive table PART is created with the value of the DBCREATE\_TABLE\_OPTS= option appended to the CREATE TABLE statement.

```
options sastrace=',,d' sastraceloc=saslog;

libname x HADOOP server=XXXX user=XXXXX pwd=XXXXXX ;
data x.part (DBCREATE_TABLE_OPTS="PARTITIONED BY(I INT)");
i=1; output;
j=2; output;
run;
```

When you use this data set option to create this table, the Hadoop interface generates a CREATE TABLE statement similar to this one.

```
HADOOP_8: Executed: on connection 2
CREATE TABLE 'PART' ('j' DOUBLE) PARTITIONED BY(I INT)
TBLPROPERTIES ('SAS OS Name'='W32_7PRO', 'SAS Version'='9.04.01M3D04152015')
```

## DBFORCE= Data Set Option

Specifies whether to force data truncation during insert processing.

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                             |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                                   |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                         |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">DBTYPE= LIBNAME option</a>                                                                                                                                                                                                                                                                                                               |

## Syntax

**DBFORCE=YES | NO**

### Syntax Description

#### YES

specifies that rows that contain data values that exceed the length of the DBMS column are inserted, and the data values are truncated to fit the DBMS column length.

#### NO

specifies that the rows that contain data values that exceed the DBMS column length are not inserted.

---

## Details

This option determines how the SAS/ACCESS engine handles rows that contain data values that exceed the length of the DBMS column. DBFORCE= works only when you create a DBMS table with DBTYPE= data set option—namely, you must specify both DBFORCE= and DBTYPE=. DBFORCE= does not work for inserts or updates. Therefore, to insert or update a DBMS table, you cannot use the DBFORCE= option—you must instead specify the options that are available with SAS procedures. For example, specify the FORCE= data set option in SAS with PROC APPEND.

FORCE= overrides DBFORCE= when you use FORCE= with PROC APPEND or the PROC SQL UPDATE statement. PROC SQL UPDATE does not warn you before it truncates data.

*Oracle:* You must set DBFORCE=YES if you use DBTYPE= to override the default data type of VARCHAR2 with NVARCHAR2 or NCHAR.

---

## Example: Truncate Data during Insert Processing

In this example, two librefs are associated with Oracle databases, and it does not specify databases and schemas because it uses the defaults. In the DATA step, MYDBLIB.DEPT is created from the Oracle data that MYORALIB.STAFF references. The LASTNAME variable is a character variable of length 20 in MYORALIB.STAFF. When MYDBLIB.DEPT is created, the LASTNAME variable is stored as a column of type character and length 10 by using DBFORCE=YES.

```
libname myoralib oracle user=tester1 password=tst1;
libname mydblib oracle user=myusr1 password=mypwd1;
data mydblib.dept(dbtype=(lastname='char(10)')
                  dbforce=yes);
  set myoralib.staff;
run;
```

---

## DBGEN\_NAME= Data Set Option

Specifies how SAS automatically renames columns (when they contain characters that SAS does not allow, such as \$) to valid SAS variable names.

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                             |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:     | DBMS                                                                                                                                                                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                  |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">DBGEN_NAME= LIBNAME option</a> , <a href="#">VALIDVARNAME= system option</a>                                                                                                                                                                                                                                                             |

---

## Syntax

**DBGEN\_NAME=DBMS | SAS**

### Syntax Description

#### **DBMS**

specifies that SAS renames DBMS columns to valid SAS variable names. SAS converts any disallowed characters to underscores. If it converts a column to a name that already exists, it appends a sequence number at the end of the new name.

#### **SAS**

specifies that SAS converts DBMS columns with disallowed characters into valid SAS variable names. SAS uses the format \_COLn, where n is the column number, starting with 0. If SAS converts a name to a name that already exists, it appends a sequence number at the end of the new name.

---

## Details

SAS retains column names when it reads data from DBMS tables unless a column name contains characters that SAS does not allow, such as \$ or @. SAS allows alphanumeric characters and the underscore (\_).

This option is intended primarily for National Language Support, notably converting kanji to English characters. English characters that are converted from kanji are often those that SAS does not allow. Although this option works for the single-byte character set (SBCS) version of SAS, SAS ignores it in the double-byte character set (DBCS) version. So if you have the DBCS version, you must first specify VALIDVARNAME=ANY before using your language characters as column variables.

---

## Example

If you specify DBGEN\_NAME=SAS, SAS renames a DBMS column named `Dept$Amt` to `_COLn`. If you specify DBGEN\_NAME=DBMS, SAS renames the `Dept$Amt` column to `Dept_Amt`.

---

## DBINDEX= Data Set Option

Detects and verifies that indexes exist on a DBMS table.

|              |                                                                                                                                                                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                |
| Defaults:    | NO [Greenplum, Impala, Microsoft SQL Server, SAP HANA]<br>LIBNAME option value [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, HAWQ, Informix, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP IQ, Teradata, Vertica, Yellowbrick] |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica, Yellowbrick                               |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                             |
| See:         | <a href="#">DBINDEX= LIBNAME option</a> , <a href="#">DBKEY= data set option</a> , <a href="#">MULTI_DATASRC_OPT= LIBNAME option</a>                                                                                                                    |
| CAUTION:     | Improper use of this option can impair performance. See “ <a href="#">MULTI_DATASRC_OPT=</a> ” on <a href="#">page 68</a> for detailed information about using this option.                                                                             |

---

## Syntax

**DBINDEX=YES | NO | <'>index-name<'>**

### Syntax Description

#### YES

triggers the SAS/ACCESS engine to search for all indexes on a table and return them to SAS for evaluation. If SAS/ACCESS finds a usable index, it passes the join WHERE clause to the DBMS for processing. A usable index should have at least the same attributes as the join column.

**NO**

indicates that no automated index search is performed.

***index-name***

verifies the index name that is specified for the index columns on the DBMS table. It requires the same type of call as when DBINDEX=YES is used.

## Details

If indexes exist on a DBMS table and are of the correct type, you can use this option to potentially improve performance when you are processing a join query.

Performance is often improved for queries that involve a large DBMS table and a relatively small SAS data set that is passed to the DBMS.

Queries must be issued to the necessary DBMS control or system tables to extract index information about a specific table or validate the index that you specified.

You can enter the DBINDEX= option as a LIBNAME option, SAS data set option, or an option with PROC SQL. Here is the order in which the engine processes it:

- 1 DATA step or PROC SQL specification.
- 2 LIBNAME statement specification

Specifying DBKEY= takes precedence over DBINDEX=.

## Examples

### Example 1

Here is the SAS data set that is used in these examples.

```
data s1;
  a=1; y='aaaaaa'; output;
  a=2; y='bbbbbb'; output;
  a=5; y='cccccc'; output;
run;
```

### Example 2: Use DBINDEX= in a SAS DATA Step

```
data a;
  set s1;
  set mydblib.dbtab(dbindex=yes) key=a;
  set mydblib.dbtab(dbindex=yes) key=a;
run;
```

The key is validated against the list from the DBMS. If a is an index, a pass-down occurs. Otherwise, the join takes place in SAS.

## Example 3: Use DBINDEX= in PROC SQL

```
proc sql;
select * from s1 aa, mydblib.dbtab(dbindex=yes) bb where aa.a=bb.a;
select * from s1 aa, mydblib.dbtab(dbindex=yes) bb where aa.a=bb.a;
/*or*/
select * from s1 aa, mydblib.dbtab(dbindex=a) bb where aa.a=bb.a;
select * from s1 aa, mydblib.dbtab(dbindex=a) bb where aa.a=bb.a;
```

## DBKEY= Data Set Option

Specifies a key column to optimize DBMS retrieval.

|              |                                                                                                                                                                                                                             |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                    |
| Category:    | Data Set Control                                                                                                                                                                                                            |
| Default:     | none                                                                                                                                                                                                                        |
| Data source: | Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p>                                     |
| See:         | <a href="#">DBINDEX= data set option</a> , <a href="#">DBNULLKEYS= data set option</a> , <a href="#">DBNULLKEYS= LIBNAME option</a>                                                                                         |
| CAUTION:     | Improper use of this option can decrease performance. For detailed information about using this option, see the <a href="#">DBINDEX= LIBNAME option</a> .                                                                   |

## Syntax

**DBKEY=(<'> *column-1*<'> <... <'> *column-n*<'> > )**

## Syntax Description

### ***column***

SAS uses this to build an internal WHERE clause to search for matches in the DBMS table based on the key column. For example:

```
select * from sas.a, dbms.b(dbkey=x) where a.x=b.x;
```

In this example, DBKEY= specifies column x, which matches the key column that the WHERE clause designates. However, if the DBKEY= column does NOT match the key column in the WHERE clause, DBKEY= is not used.

## Details

### Overview

You can use this option to potentially improve performance when you are processing a join that involves a large DBMS table and a small SAS data set or DBMS table.

When you specify DBKEY=, it is *strongly* recommended that an index exists for the key column in the underlying DBMS table. Performance can be severely degraded without an index.

### Interaction When DBNULLKEYS=YES

When DBNULLKEYS= is specified as YES (or is YES by default), but a particular column has been specified as NOT NULL in DB2, any comparison in the WHERE clause must explicitly exclude SAS missing values. Otherwise, rows in a SAS table with missing values could incorrectly match rows with zeros in a DB2 table.

For example, suppose you specify the Age variable in DB2 as NOT NULL in two tables, A and B. Check for missing values of Age when you use it in a WHERE clause.

```
select * from sas.a, dbms.b(dbkey=(name address age))
  where a.name=b.name and
        a.address=b.address and
        a.age=b.age and a.age is not null;
```

---

## Examples

### Example 1: Using DBKEY= with MODIFY=

This example uses DBKEY= with the MODIFY statement in a DATA step:

```
libname invty db2;
data invty.stock;
  set addinv;
  modify invty.stock(dbkey=partno) key=dbkey;
  INSTOCK=instock+nwstock;
  RECDATE=today();
  if _iorc_=0 then replace;
run;
```

### Example 2: Using More Than One DBKEY= Value

To use more than one value for DBKEY=, you must include the second value as a join on the WHERE clause. In the next example, PROC SQL brings the entire DBMS table into SAS and then proceeds with processing:

```
options sastrace=',,d' sastraceloc=saslog nostsuffix;
proc sql;
```

```

create table work.barbkey as
  select keyvalues.empid, employees.hiredate, employees.jobcode
    from mydblib.employees(dbkey=(empid jobcode))
  inner join work.keyvalues on employees.empid = keyvalues.empid;
quit;

```

## DBLABEL= Data Set Option

Specifies whether to use SAS variable labels or SAS variable names as the DBMS column names during output processing.

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                             |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                                   |
| Restriction: | This option is valid only for creating DBMS tables.                                                                                                                                                                                                                                                                                                  |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, HAWQ, Informix, Impala, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                          |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |

## Syntax

**DBLABEL=YES | NO**

### Syntax Description

#### YES

specifies that SAS variable *labels* are used as DBMS column names during output processing.

#### NO

specifies that SAS variable *names* are used as DBMS column names.

## Example: Specify a Variable Label

In this example, a SAS data set, NEW, is created with one variable C1. This variable is assigned a label of DEPTNUM. In the second DATA step, the MYDBLIB.MYDEPT

table is created by using DEPTNUM as the DBMS column name. When you specify DBLABEL=YES, the label can be used as the column name.

```

data new;
  label c1='deptnum';
  c1=001;
run;
data mydblib.mydept(dblabel=yes);
  set new;
run;
proc print data=mydblib.mydept;
run;
```

## DBLINK= Data Set Option

Specifies a link from your local database to database objects on another server [Oracle]. Specifies a link from your default database to another database on the server to which you are connected [SAP ASE].

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Default:     | LIBNAME option value                                                     |
| Data source: | Oracle, SAP ASE                                                          |
| See:         | <a href="#">DBLINK= LIBNAME option</a>                                   |

## Syntax

**DBLINK=***database-link*

## Details

This option operates differently in each DBMS.

**Oracle:** A link is a database object that identifies an object that is stored in a remote database. A link contains stored path information and can also contain user name and password information for connecting to the remote database. If you specify a link, SAS uses the link to access remote objects. If you omit DBLINK=, SAS accesses objects in the local database.

**SAP ASE:** You can use this option to link to another database within the same server to which you are connected. If you omit DBLINK=, SAS can access objects only in your default database.

## Example: Specify an Oracle Link

In this example, SAS sends MYORADB.EMPLOYEES to Oracle as EMPLOYEES@SALES.HQ.ACME.COM.

```
proc print data=myoradb.employees(dblink='sales.hq.acme.com');
run;
```

---

## DBLARGETABLE= Data Set Option

Designates which table is the larger table when you are processing a join that involves tables from two different types of databases.

|              |                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                            |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                    |
| Alias:       | DBMASTER=                                                                                                                                                                                                                                                                                                                                           |
| Default:     | none                                                                                                                                                                                                                                                                                                                                                |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                 |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">MULTI_DATASRC_OPT= LIBNAME option</a>                                                                                                                                                                                                                                                                                                   |

---

## Syntax

**DBLARGETABLE=YES**

### Syntax Description

#### YES

designates which of two tables that are referenced in a join operation is the larger table.

## Details

You can use this option to specify which table reference in a join is the larger table. This can improve performance by eliminating the processing that is normally performed to determine this information. However, this option is ignored when outer joins are processed.

## Example: Join Two Tables

In this example, a table from an Oracle database and a table from a DB2 database are joined. DBLARGETABLE= is set to YES to indicate that the Oracle table is the larger table. The DB2 table is the smaller table.

```
libname mydblib oracle user=myusr1 /*database 1 */
      pw=mypwd1 path='myorapath';
libname mydblib2 db2 user=myusr1 /*database 2 */
      pw=mypwd1 path='mydb2path';
proc sql;
  select * from mydblib.bigtab(dblargetable=yes), mydblib2.smalltab
  where bigtab.x=smalltab.x;
quit;
```

## DBMAX\_TEXT= Data Set Option

Determines the length of very large DBMS character data types, such as BLOB or CLOB, that are read into SAS or written from SAS when you are using a SAS/ACCESS engine.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Alias: TEXTSIZE=

Default: 1024

Restrictions: This option applies when you retrieve, append, and update rows in an existing table. It does not apply when you create a table.

This option is ignored for CHAR, VARCHAR, and VARCHAR2 (Oracle) data types.

Requirements: Specify this value as 4000 when you are using procedures that work with SAS High-Performance Analytics Server. [Oracle]

Specify this value as 4000 or lower when you are using procedures that work with the SAS High-Performance Analytics Server. [Amazon Redshift, DB2, Hadoop, Impala, JDBC, Microsoft SQL Server, ODBC, PostgreSQL, SAP HANA, Spark]

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, Hadoop, HAWQ, Impala, JDBC, MySQL, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for JDBC was added in SAS Viya 3.4.

The number of bytes that are used to store characters might vary and is based on your session encoding.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Spark was added in SAS 9.4M7.

Support for Yellowbrick was added in SAS 9.4M7.

See: [DBMAX\\_TEXT= LIBNAME option](#)

## Syntax

**DBMAX\_TEXT=***integer*

### Syntax Description

***integer***

is a number between 1 and 32,767.

## Details

In general, this option applies only to BLOB, CLOB, and other large object or very long DBMS character data types, such as the SAP ASE TEXT data type. Also, remember that the number of bytes that are used to store characters might vary and is based on your session encoding.

If the way that you specify the value of DBMAX\_TEXT= truncates data in a table, the data load fails for that table.

*Hadoop*: This option applies for the STRING data type.

*Oracle*: For SAS 9 or higher, this option applies for CLOB, BLOB, LONG, LONG RAW, and LOB data types. The behavior of the ACCESS and DBLOAD procedures has not changed since SAS 8. So only LONG and LOB data types are valid if you use this option with those procedures.

## DBNULL= Data Set Option

Indicates whether NULL is a valid value for the specified columns when a table is created.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: \_ALL\_=YES. (All columns are created without the NOT NULL constraint. That is, all columns allow NULL values.)

Restriction: This option is valid for the Hadoop engine only when the HDMD\_METADIR= connection option is not set.

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick

- Notes: Support for HAWQ was added in SAS 9.4M3.  
 Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.  
 Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.  
 Support for Yellowbrick was added in SAS 9.4M7.
- See: [NULLCHAR= data set option](#), [NULLCHARVAL= data set option](#)

## Syntax

**DBNULL=<(>column-name-1=YES | NO< column-name-2=YES | NO...><)>**  
**DBNULL=\_ALL\_=YES | NO**

## Syntax Description

### **\_ALL\_**

specifies that the YES or NO applies to all columns in the table. You can specify `_ALL_=YES` or `_ALL_=NO` in combination with one or more columns.

### **YES**

specifies that the NULL value is valid for the specified column in the DBMS table.

### **NO**

specifies that the NULL value is not valid for the specified column in the DBMS table.

## Details

This option is valid only for creating DBMS tables. If you specify more than one column name, you must separate them with spaces and include the list in parentheses.

The DBNULL= option processes values from left to right. If you specify a column name twice or if you use the \_ALL\_ value, the last value overrides the first value that you specified for the column.

## Examples

### Example 1: Prevent Specific Columns from Accepting Null Values

In this example, you can use the DBNULL= option to prevent the EMPID and JOBCODE columns in the new MYDBLIB.MYDEPT2 table from accepting null values. If the EMPLOYEES table contains null values in the EMPID or JOBCODE columns, the DATA step fails.

```
data mydblib.mydept2 (dbnull=(empid=no jobcode=no)) ;
```

```
      set mydblib.employees;
run;
```

## Example 2: Prevent All Columns from Accepting Null Values

In this example, all columns in the new MYDBLIB.MYDEPT3 table except for the JOBCODE column are prevented from accepting null values. If the EMPLOYEES table contains null values in any column other than the JOBCODE column, the DATA step fails.

```
data mydblib.mydept3(dbnull=(_ALL_=no jobcode=YES));
      set mydblib.employees;
run;
```

## DBNULLKEYS= Data Set Option

Controls the format of the WHERE clause with regard to NULL values when you use the DBKEY= data set option.

|              |                                                                                                                                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                   |
| Category:    | Data Set Control                                                                                                                                                                                                           |
| Default:     | LIBNAME option value                                                                                                                                                                                                       |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p>                                    |
| See:         | <a href="#">DBINDEX= data set option</a> , <a href="#">DBKEY= data set option</a> , <a href="#">DBNULLKEYS= LIBNAME option</a>                                                                                             |

## Syntax

**DBNULLKEYS=YES | NO**

## Required Arguments

### YES

specifies that the key columns in a transaction table or a master table might contain NULL values.

### NO

specifies that the key columns in a transaction table or a master table do not contain NULL values.

## Details

If there might be NULL values in the transaction table or the master table for the columns that you specify in the DBKEY= option, use DBNULLKEYS=YES. When you specify DBNULLKEYS=YES and specify a column that the DBKEY= data set option defines as NOT NULL, SAS generates a WHERE clause to find NULL values. For example, if you specify DBKEY=COLUMN and COLUMN is not specified as NOT NULL, SAS generates a WHERE clause with this syntax:

```
WHERE ((COLUMN = ?) OR ((COLUMN IS NULL) AND (? IS NULL)))
```

This syntax enables SAS to prepare the statement once and use it for any value (NULL or NOT NULL) in the column. This syntax has the potential to be much less efficient than the shorter form of the following WHERE clause. When you specify DBNULLKEYS=NO or specify a column that is specified as NOT NULL in the DBKEY= option, SAS generates a simple WHERE clause.

If you know that there are no NULL values in the transaction table or the master table for the columns that you specify in the DBKEY= option, you can use DBNULLKEYS=NO. If you specify DBNULLKEYS=NO and DBKEY=COLUMN, SAS generates a shorter form of the WHERE clause, regardless of whether the column that is specified in DBKEY= is specified as NOT NULL.

```
WHERE (COLUMN = ?)
```

---

## DBNULLWHERE= Data Set Option

Specifies whether character columns in a WHERE clause can contain NULL values.

|              |                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                 |
| Category:    | Data Set Control                                                                                         |
| Default:     | LIBNAME option value                                                                                     |
| Interaction: | If the DBNULL= data set option is specified, then the value of DBNULLWHERE= is automatically set to YES. |
| Data source: | DB2 under UNIX and PC Hosts, JDBC, Microsoft SQL Server, ODBC, Oracle, SAP HANA                          |
| Notes:       | Support for this data set option was added in SAS 9.4M5.<br>Support for JDBC was added in SAS Viya 3.4.  |
| See:         | <a href="#">DBNULL= data set option</a> , <a href="#">DBNULLWHERE= LIBNAME option</a>                    |

---

## Syntax

**DBNULLWHERE=YES | NO**

## Required Arguments

### **YES**

specifies that there might be a NULL value for a column that is listed in a WHERE clause.

### **NO**

specifies that none of the columns in a WHERE clause contain NULL values.

## Details

This option applies to character columns only.

When DBNULLWHERE=YES, SAS/ACCESS verifies whether blank or NULL values are possible for each character column that you include in a WHERE clause.

When DBNULLWHERE=NO, SAS/ACCESS does not check to see whether NULL values are possible for the specified columns in a WHERE clause. When you know that none of your specified columns contain NULL values, specifying DBNULLWHERE=NO can result in a faster query because fewer conditions are being checked.

## DBPROMPT= Data Set Option

Specifies whether SAS displays a window in SAS Display Manager that prompts you to enter DBMS connection information.

|              |                                                                                                                                                                                                                                                            |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                   |
| Category:    | Data Set Control                                                                                                                                                                                                                                           |
| Defaults:    | LIBNAME option value [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, Impala, Microsoft SQL Server, MySQL, Netezza, ODBC, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Yellowbrick]<br>NO [SAP ASE, Vertica]                                   |
| Restriction: | This option is not applicable to SAS Viya.                                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Microsoft SQL Server, MySQL, Netezza, ODBC, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick                                                           |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">DBPROMPT= LIBNAME option</a>                                                                                                                                                                                                                   |

## Syntax

**DBPROMPT=YES | NO**

## Syntax Description

### **YES**

displays the prompting window.

### **NO**

does not display the prompting window.

## Details

This data set option is supported only for view descriptors.

*Oracle:* You can enter 30 characters in the Oracle interface each for the user name and password and up to 70 characters for the path, depending on your platform and terminal type.

## Examples

### Example 1: Use the Default Value (No Prompt)

In this example, connection information is specified in the ACCESS procedure. The DBPROMPT= data set option defaults to NO during the PRINT procedure because it is not specified.

```
proc access dbms=oracle;
  create alib.mydesc.access;
  user=myusr1;
  password=mypwd1;
  table=dept;
  create vlib.myview.view;
  select all;
run;
proc print data=vlib.myview;
run;
```

### Example 2: Prompt for Connection Information

In the next example, the DBPROMPT window appears during connection to the DBMS. Values that were previously specified during the creation of MYVIEW are pulled into the DBPROMPT window fields. You must edit or accept the connection information in the DBPROMPT window to proceed. The password value appears as a series of asterisks; you can edit it.

```
proc print data=vlib.myview(dbprompt=yes) ;
run;
```

---

## DBSASLABEL= Data Set Option

Specifies how the engine returns column labels.

|              |                                                                                                                                                                                                                                                                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                              |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                      |
| Default:     | COMPAT                                                                                                                                                                                                                                                                                                                                                |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick                                                                             |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Hadoop and JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Spark and Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">DBSASLABEL= LIBNAME option</a>                                                                                                                                                                                                                                                                                                            |

---

## Syntax

**DBSASLABEL=**[COMPAT | DBMS | NONE](#)

### Syntax Description

#### **COMPAT**

specifies that the labels returned should be compatible with what the application normally receives. In other words, engines exhibit their normal behavior.

#### **DBMS**

specifies that the engine returns a label exactly as it is stored in the database.

Supports SAP HANA

#### **NONE**

specifies that the engine does not return a column label. The engine returns blanks for the column labels.

---

## Details

By default, the SAS/ACCESS interface for your DBMS generates column labels from column names instead of from the real column labels.

You can use this option to override the default behavior. It is useful for when PROC SQL uses column labels as headings instead of column aliases.

## Example: Return Blank Labels for Aliases in Headings

This example shows how to use DBSASLABEL= to return blank column labels so that PROC SQL can use the column aliases as the column headings.

```
proc sql;
    select deptno as Department ID, loc as Location
    from mylib.dept (dbsaslabel=none);
```

When DBSASLABEL=NONE, PROC SQL ignores the aliases, and it uses DEPTNO and LOC as column headings in the result set.

## DBSASTYPE= Data Set Option

Specifies data types to override the default SAS data types during input processing.

|              |                                                                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                             |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                     |
| Default:     | DBMS-specific                                                                                                                                                                                                                                                                                                                                        |
| Restriction: | The Snowflake interface does not support UPDATE operations or using the DATA step MODIFY statement due to Snowflake client and database limitations.                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                                                           |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |

## Syntax

**DBSASTYPE=(***column-name-1=<'> SAS-data-type<'>*  
*<...column-name-n=<'> SAS-data-type<'> > )*

### Syntax Description

#### ***column-name***

specifies a DBMS column name.

**SAS-data-type**

specifies a SAS data type, which can be CHAR(*n*), NUMERIC, DATETIME, DATE, TIME. See the DBMS-specific reference section for your SAS/ACCESS interface for details.

## Details

By default, the SAS/ACCESS interface for your DBMS converts each DBMS data type to a SAS data type during input processing. When you need a different data type, you can use this option to override the default and assign a SAS data type to each specified DBMS column. Some conversions might not be supported. In that case, SAS prints an error to the log.

If you convert a long string value to the NUMERIC type, the numeric value that is stored in SAS might not exactly match the original character value. This happens with long strings that contain more than 15 significant digits. For example, if SAS reads in a character value of '123456789012345678901234567890' and converts that to type NUMERIC, then the numeric value that SAS stores is 12345678901234600000000000000. For more information, see “[Choosing Your Degree of Numeric Precision](#)” on page 8.

## Examples

### Example 1: Override the Default Data Type

In this example, DBSASTYPE= specifies a data type to use for the MYCOLUMN column when SAS prints ODBC data. SAS can print the values if the data in this DBMS column is stored in a format that SAS does not support, such as SQL\_DOUBLE(20).

```
proc print data=mylib.mytable
  (dbsastype=(mycolumn='CHAR(20)'));
run;
```

### Example 2: Convert Column Length

In the next example, data that is stored in the DBMS FIBERSIZE column has a data type that provides more precision than SAS can accurately support, such as DECIMAL(20). If you use only PROC PRINT on the DBMS table, the data might be rounded or displayed as a missing value. So you could use DBSASTYPE= instead to convert the column so that the length of the character field is 21. The DBMS performs the conversion before the data is brought into SAS, so precision is preserved.

```
proc print data=mylib.specprod
  (dbsastype=(fibersize='CHAR(21)'));
run;
```

## Example 3: Append Tables to Match Data Types

The next example uses DBSASTYPE= to append one table to another when the data types cannot be compared. If the EMPID variable in the SAS data set is specified as CHAR(20) and the EMPID column in the DBMS table is specified as DECIMAL(20), you can use DBSASTYPE= to make them match:

```
proc append base=dblib.hrdata (dbsastype=(empid='CHAR(20)'))
    data=saslib.personnel;
run;
```

DBSASTYPE= specifies to SAS that the EMPID is specified as a character field of length 20. When a row is inserted from the SAS data set into a DBMS table, the DBMS performs a conversion of the character field to the DBMS data type DECIMAL(20).

## DBSLICE= Data Set Option

Specifies user-supplied WHERE clauses to partition a DBMS query for threaded Reads.

|              |                                                                                                                                                                                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                   |
| Category:    | Data Set Control                                                                                                                                                                                                                                                           |
| Default:     | none                                                                                                                                                                                                                                                                       |
| Data source: | DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Informix, Microsoft SQL Server, ODBC, Oracle, SAP ASE, SAP HAHA, SAP IQ, Teradata, Vertica                                                                                                                   |
| Notes:       | <p>Support for Vertica was added for SAS 9.4.</p> <p>Support for SAP HANA was added in SAS 9.4M1.</p> <p>Support for Greenplum was added in SAS 9.4M2.</p> <p>Support for HAWQ was added in SAS 9.4M3.</p>                                                                 |
| See:         | <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a>                                                                                                                                                                                 |
| CAUTION:     | <p><b>When using DBSLICE=, you are responsible for data integrity.</b> If your WHERE clauses omit rows from the result set or retrieve the same row on more than one thread, your input DBMS result set is incorrect and your SAS program generates incorrect results.</p> |

## Syntax

**DBSLICE=(**"*WHERE-clause-1*" "*WHERE-clause-2*" <..." *WHERE-clause-n*"**)**

**DBSLICE=(**<*server=>* "*WHERE-clause-1*" <*server=>* "*WHERE-clause-2*" < ... <*server=>* "*WHERE-clause-n*"**)**

## Syntax Description

### **WHERE-clause**

The WHERE clauses in the syntax signify DBMS-valid WHERE clauses that partition the data. The clauses should not cause any omissions or duplications of rows in the results set. For example, if EMPNUM can be null, this DBSLICE= specification omits rows, creating an *incorrect* result set:

```
DBSLICE=("EMPNUM<1000" "EMPNUM>=1000")
```

Here is a correct form:

```
DBSLICE=("EMPNUM<1000" "EMPNUM>=1000" "EMPNUM IS NULL")
```

In this example, DBSLICE= creates an *incorrect* set by duplicating SALES with a value of 0.

```
DBSLICE=("SALES<=0 or SALES=NULL" "SALES>=0")
```

#### **server**

identifies a particular server node in a DB2 partitioned database or in a Microsoft SQL Server partitioned view. Use this to obtain the best possible Read performance so that your SAS thread can connect directly to the node that contains the data partition that corresponds to your WHERE clause. For DBMS-specific details, see [DB2 under UNIX and PC Hosts on page 761](#) and [ODBC on page 1087](#).

## Details

If your table reference is eligible for threaded Reads (that is, if it is a read-only LIBNAME table reference), DBSLICE= forces a threaded Read to occur. This partitions the table with the WHERE clauses that you provide. Use DBSLICE= when SAS is unable to generate threaded Reads automatically, or if you can provide better partitioning.

DBSLICE= is appropriate for experienced programmers familiar with the layout of their DBMS tables. A well-tuned DBSLICE= specification usually outperforms SAS automatic partitioning. For example, a well-tuned DBSLICE= specification might better distribute data across threads by taking advantage of a column that SAS/ACCESS cannot use when it automatically generates partitioning WHERE clauses.

DBSLICE= delivers optimal performance for DB2 under UNIX and for Microsoft SQL Server. Conversely, DBSLICE= can degrade performance compared to automatic partitioning. For example, Teradata starts the FastExport Utility for automatic partitioning. If DBSLICE= overrides this action, WHERE clauses are generated instead. Even with well planned WHERE clauses, performance is degraded because FastExport is considerably faster.

## Examples

### Example 1: Partition a Column (Two Threads)

In this example, DBSLICE= partitions on the GENDER column can have only the values **m**, **M**, **f**, and **F**. This DBSLICE= clause does not work for all DBMSs due to the use of UPPER and single quotation marks. Some DBMSs require double quotation marks around character literals. Two threads are created.

```
proc reg SIMPLE
data=lib.customers(DBSLICE="UPPER(GENDER)='M'" "UPPER(GENDER)='F'") ;
var age weight;
where years_active>1;
```

```
run;
```

## Example 2: Partition a Column (Three Threads)

The next example partitions on the non-null column CHILDREN, the number of children in a family. Three threads are created.

```
data local;
set lib.families(DBSLICE=( "CHILDREN<2" "CHILDREN>2" "CHILDREN=2" )) ;
where religion="P";
run;
```

## DBSLICEPARM= Data Set Option

Controls the scope of DBMS threaded Reads and the number of DBMS connections.

|              |                                                                                                                                                                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC Steps (when accessing DBMS data using SAS/ACCESS software)<br>(also available as a SAS configuration file option, SAS invocation option, global SAS option, and LIBNAME option)                                                                                                               |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                            |
| Defaults:    | NONE [DB2 under UNIX and PC Hosts, Greenplum, Microsoft SQL Server, Vertica]<br>THREADED_APPS, none [HAWQ]<br>THREADED_APPS,2 [DB2 under z/OS, Oracle, and Teradata]<br>THREADED_APPS,2 or THREADED_APPS,3 [Informix, ODBC, SAP ASE, SAP HANA, SAP IQ]                                                      |
| Data source: | DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Informix, Microsoft SQL Server, ODBC, Oracle, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica                                                                                                                                                    |
| Notes:       | Support for Vertica was added for SAS 9.4.<br>Support for SAP HANA was added in SAS 9.4M1.<br>Support for Greenplum was added in SAS 9.4M2.<br>Support for HAWQ was added in SAS 9.4M3.                                                                                                                     |
| See:         | <a href="#">DBSLICE= data set option</a> , <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">SLEEP= LIBNAME option</a> , <a href="#">SLEEP= data set option</a> , <a href="#">TENACITY= LIBNAME option</a> , <a href="#">TENACITY= data set option</a> |

## Syntax

```
DBSLICEPARM=NONE | THREADED_APPS | ALL
DBSLICEPARM=( NONE | THREADED_APPS | ALL <max-threads> )
DBSLICEPARM=( NONE | THREADED_APPS | ALL< , max-threads > )
```

## Syntax Description

### **NONE**

disables DBMS threaded Reads. SAS reads tables on a single DBMS connection, as it did with SAS 8 and earlier.

### **THREADED\_APPS**

makes fully threaded SAS procedures (threaded applications) eligible for threaded Reads.

### **ALL**

makes all read-only librefs eligible for threaded Reads. It includes SAS threaded applications, the SAS DATA step, and numerous SAS procedures.

### **max-threads**

a positive integer value that specifies the maximum number of connections per table read. The second parameter of the option determines the number of threads to read the table in parallel. The number of partitions on the table determine the number of connections made to the Oracle server for retrieving rows from the table. A partition or portion of the data is read on each connection. The combined rows across all partitions are the same regardless of the number of connections. That is, changes to the number of connections do not change the result set. Increasing the number of connections instead redistributes the same result set across more connections.

There are diminishing returns when increasing the number of connections. With each additional connection, more burden is placed on the DBMS, and a smaller percentage of time is saved in SAS. See the DBMS-specific reference section about threaded Reads for your interface before using this parameter.

## Details

You can use DBSLICEPARM= in numerous locations. The usual rules of option precedence apply: A table (data set) option has the highest precedence, followed by a LIBNAME option, and so on. A SAS configuration file option has the lowest precedence because DBSLICEPARM= in any of the other locations overrides that configuration value.

DBSLICEPARM=ALL and DBSLICEPARM=THREADED\_APPS make SAS programs eligible for threaded Reads. To determine whether threaded Reads are actually generated, turn on SAS tracing and run a program, as shown in this example.

```
options sastrace=",,d" sastraceloc=saslog nostsuffix;
proc print data=lib.dbtable(dbsliceparm=(ALL));
  where dbcol>1000;
run;
```

If you want to directly control the threading behavior, use the DBSLICE= data set option.

*Greenplum, HAWQ:* There is no default value for the maximum number of connections per table read. This value depends on the number of partitions in a table and the arguments that are used with the MOD function in a WHERE clause. For more information, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page [76](#).

DB2 under UNIX and PC Hosts, Informix, Microsoft SQL Server, ODBC, SAP ASE, SAP IQ: The default thread number depends on whether an application passes in the number of threads (CPUCOUNT=) and whether the data type of the column that was selected for purposes of data partitioning is binary.

## Examples

### Example 1: Disable Threaded Reads for All SAS Users

Here is how to use DBSLICEPARM= in a SAS configuration file entry in Windows to turn off threaded Reads for all SAS users.

```
-dbsliceparm NONE
```

### Example 2: Enable Threaded Reads for Read-Only References

Here is how you can use DBSLICEPARM= as a z/OS invocation option to turn on threaded Reads for read-only references to DBMS tables throughout a SAS job.

```
sas o(dbsliceparm=ALL)
```

### Example 3: Increase Maximum Threads (as a SAS Global Option)

In this example, you can use DBSLICEPARM= as a SAS global option to increase maximum threads to three for SAS threaded applications. Most likely, you would use it as one of the first statements in your SAS code.

```
option dbsliceparm=(threaded_apps,3);
```

### Example 4: Enable Threaded Reads for References Using a Particular Libref

You can use DBSLICEPARM= as a LIBNAME option to turn on threaded Reads for read-only table references that use this particular libref, as shown in this example

```
libname dblib oracle user=myusr1 password=mypwd1 dbsliceparm=ALL;
```

### Example 5: Enable Threaded Reads as a Table-Level Option

Here is how to use DBSLICEPARM= as a table-level option to turn on threaded Reads for this particular table, requesting up to four connections.

```
proc reg SIMPLE;
  data=dblib.customers (dbsliceparm=(all,4));
```

```

var age weight;
  where years_active>1;
run;

```

## DBTYPE= Data Set Option

Specifies a data type to use instead of the default DBMS data type when SAS creates a DBMS table.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: DBMS-specific

Restriction: You cannot use this option with the HDFS\_METADIR= connection option.

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for JDBC was added in SAS Viya 3.4.

Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

See: [DBCREATE\\_TABLE\\_OPTS= data set option](#), [DBFORCE= data set option](#), [DBNULL= data set option](#)

## Syntax

**DBTYPE=(***column-name-1=<'> DMBS-type<'>*  
*<...column-name-n=<'> DMBS-type<'> > )*

## Syntax Description

### ***column-name***

specifies a DBMS column name.

### ***DMBS-type***

specifies a DBMS data type. See the DBMS-specific reference section for your SAS/ACCESS interface for the default data types for your DBMS.

## Details

By default, the SAS/ACCESS interface for your DBMS converts each SAS data type to a predetermined DBMS data type when it writes data to your DBMS. When you

need a different data type, use DBTYPE= to override the default data type chosen by the SAS/ACCESS engine.

You can also use this option to specify column modifiers. The allowable syntax for these modifiers is generally DBMS-specific. For more information, see the SQL reference for your database.

*Google BigQuery:* The following table shows the conversion from the data type that you specify for DBTYPE= to the resulting Google BigQuery data type when you create a table in the DBMS.

**Table 13.5** Conversion from DBTYPE= to Google BigQuery Data Types

| Data Type Specified for DBTYPE= | Resulting Google BigQuery Data Type |
|---------------------------------|-------------------------------------|
| BIGINT                          | INT64                               |
| DATE                            | DATE                                |
| DOUBLE                          | FLOAT64                             |
| TIME                            | TIME                                |
| TIMESTAMP                       | TIMESTAMP                           |
| VARBINARY                       | BYTES                               |
| VARCHAR                         | STRING                              |

*MySQL:* All text strings are passed as is to the MySQL server. MySQL truncates text strings to fit the maximum length of the field without generating an error message.

*Teradata:* You can use DBTYPE= to specify data attributes for a column. See your Teradata CREATE TABLE documentation for information about the data type attributes that you can specify. If you specify DBNULL=NO for a column, do not also use DBTYPE= to specify NOT NULL for that column. If you do, NOT NULL is inserted twice in the column definition. This causes Teradata to generate an error message.

*Vertica:* The default is none.

## Examples

### Example 1: Specify Data Types for Columns

In this example, DBTYPE= specifies the data types to use when you create columns in the DBMS table.

```
data mydblib.newdept(dbtype=(deptno='number(10,2)' city='char(25)'));
      set mydblib.dept;
run;
```

## Example 2: Specify Data Types for Columns in a New Table

This example creates a new Teradata table, Newdept, specifying the Teradata data types for the DEPTNO and CITY columns.

```
data mydblib.newdept (dbtype=(deptno='byteint' city='char(25)')) ;
set dept;
run;
```

## Example 3: Specify a Data Type for a Column in a New Table

This example creates a new Teradata table, Newemployees, and specifies a data type and attributes for the EMPNO column. The example encloses the Teradata type and attribute information in double quotation marks. Single quotation marks conflict with single quotation marks that the Teradata FORMAT attribute requires. If you use single quotation marks, SAS returns syntax error messages.

```
data mydblib.newemployees (dbtype= (empno="SMALLINT FORMAT '9(5)' 
CHECK (empno >= 100 AND empno <= 2000)) ;
set mydblib.employees;
run;
```

## Example 4: Create a Primary Key for a New Table

Where x indicates the Oracle engine, this example creates a new table, Allacctx, and uses DBTYPE= to create the primary key, Allacct\_Pk.

```
data x.ALLACCTX ( dbtype=
SourceSystem = 'varchar(4)'
acctnum = 'numeric(18,5) CONSTRAINT "ALLACCT_PK" PRIMARY KEY'
accttype = 'numeric(18,5)'
balance = 'numeric(18,5)'
clientid = 'numeric(18,5)'
closedate = 'date'
opendate = 'date'
primary_cd = 'numeric(18,5)'
status = 'varchar(1)'
);
set work.ALLACCT ;
format CLOSEDATE date9. ;
format OPENDATE date9. ;
run;
```

The code generates this CREATE TABLE statement.

**Output 13.1** Output from a CREATE TABLE Statement That Uses DBTYPE= to Specify a Column Modifier

```
CREATE TABLE ALLACCTX(SourceSystem varchar(4),
acctnum numeric(18,5) CONSTRAINT "ALLACCT_PK" PRIMARY KEY,
accttype numeric(18,5),balance numeric(18,5),clientid numeric(18,5),
closedate date,opendate date,primary_cd numeric(18,5),status varchar(1))
```

## Example 5: Data Type Conversions When Creating a Table in Google BigQuery

Given that x indicates a Google BigQuery libref, the following DATA step creates the table Mytest in the Google BigQuery database.

```
data x.mytest(dbtype=(icol1='bigint' charcol1='varchar(20)'));
      icol1=1; charcol1='test1';
      run;
```

The resulting table, Mytest, is created where column Icol1 is of type INT64 and column Charcol1 is of type STRING. For more information, see the information about Google BigQuery in the Details section above.

## DEGREE= Data Set Option Data Set Option

Determines whether DB2 uses parallelism.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Default:     | ANY                                                                      |
| Data source: | DB2 under z/OS                                                           |
| See:         | <a href="#">DEGREE= LIBNAME option</a>                                   |

## Syntax

**DEGREE=**[ANY](#) | [1](#)

### Syntax Description

#### **ANY**

enables DB2 to use parallelism, and issues the SET CURRENT DEGREE ='xxx' for all DB2 threads that use that libref.

#### **1**

explicitly disables the use of parallelism.

## Details

When DEGREE=ANY, DB2 has the option of using parallelism when it is appropriate.

Specifying DEGREE=1 prevents DB2 from performing parallel operations. Instead, DB2 is restricted to performing one task that, although this is perhaps slower, it uses fewer system resources.

---

## DIMENSION= Data Set Option

Specifies whether the database creates dimension tables or fact tables.

|              |                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                   |
| Category:    | Data Set Control                                                                                                                           |
| Default:     | NO                                                                                                                                         |
| Data source: | Aster                                                                                                                                      |
| See:         | <a href="#">DIMENSION= LIBNAME option</a> , <a href="#">PARTITION_KEY= LIBNAME option</a> , <a href="#">PARTITION_KEY= data set option</a> |

---

## Syntax

**DIMENSION=YES | NO**

### Syntax Description

#### YES

specifies that the database creates dimension tables.

#### NO

specifies that the database creates fact tables.

---

## DISTRIBUTE\_ON= Data Set Option

Specifies one or more column names to use in the DISTRIBUTE ON clause of the CREATE TABLE statement.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Alias:       | DISTRIBUTE=                                                              |
| Default:     | none                                                                     |
| Data source: | Netezza                                                                  |

---

## Syntax

**DISTRIBUTE\_ON='column-1<...,column-n>' | RANDOM**

## Syntax Description

### **column-name**

specifies one or more DBMS column names. Separate multiple column names with commas.

### **RANDOM**

specifies that data is distributed evenly. The Netezza Performance Server does this across all SPUs. This is known as *round-robin distribution*.

## Details

You can use this option to specify a column name to use in the DISTRIBUTE ON= clause of the CREATE TABLE statement. Each table in the database must have a distribution key that consists of one to four columns. If you do not specify this option, the DBMS selects a distribution key.

## Examples

### Example 1: Create a Distribution Key on a Single Column

```
proc sql;
  create table netlib.customtab(DISTRIBUTE_ON='partno')
    as select partno, customer, orderdat from saslib.orders;
quit;
```

### Example 2: Create a Distribution Key on Many Columns

For more than one column, separate the columns with commas.

```
data netlib.mytab(DISTRIBUTE_ON='col1,col2');
  col1=1;col2=12345;col4='mytest';col5=98.45;
run;
```

### Example 3: Use the RANDOM Keyword

```
data netlib.foo(distribute_on=random);
  mycol1=1;mycol2='test';
run;
```

---

## DISTRIBUTED\_BY= Data Set Option

Uses one or multiple columns to distribute table rows across database segments.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Default:     | RANDOMLY DISTRIBUTED                                                     |
| Data source: | Greenplum, HAWQ                                                          |
| Note:        | Support for HAWQ was added in SAS 9.4M3.                                 |

---

## Syntax

**DISTRIBUTED\_BY='column-1 <...,column-n>' | DISTRIBUTED RANDOMLY**

### Syntax Description

#### *column-name*

specifies a DBMS column name.

#### **DISTRIBUTED RANDOMLY**

determines the column or set of columns that the Greenplum database uses to distribute table rows across database segments. This is known as round-robin distribution.

---

## Details

For uniform distribution—namely, so that table records are stored evenly across segments (machines) that are part of the database configuration—the distribution key should be as unique as possible.

---

## Example: Create a Table By Specifying a Distribution Key

```
libname x greenplm user=myusr1 password=mypwd1 dsn=mysrv1;
data x.sales (dbtype=(id=int qty=int amt=int)
               distributed_by='distributed by (id)');
  id = 1;
  qty = 100;
  sales_date = '27Aug2009'd;
  amt = 20000;
run;
```

It creates the SALES table.

```
CREATE TABLE SALES
(id int,
qty int,
sales_date double precision,
amt int
) distributed by (id)
```

## ERRLIMIT= Data Set Option

Specifies the number of errors that are allowed before SAS stops processing and issues a rollback.

|              |                                                                                                                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                          |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                  |
| Default:     | 1                                                                                                                                                                                                                                                                                                 |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                    |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">DBCOMMIT= LIBNAME option</a> , <a href="#">DBCOMMIT= data set option</a> , <a href="#">ERRLIMIT= LIBNAME option</a> , <a href="#">ML_CHECKPOINT= data set option</a>                                                                                                                  |

## Syntax

**ERRLIMIT=***integer*

### Syntax Description

#### *integer*

specifies a positive integer that represents the number of errors after which SAS stops processing and issues a rollback.

## Details

### General Information about ERRLIMIT=

SAS ends the step abnormally and calls the DBMS to issue a rollback after a specified number of errors while processing inserts, deletes, updates, and appends. If ERRLIMIT=0, SAS processes all rows no matter how many errors occur. The SAS log displays the total number of rows that SAS processed and the number of failed rows, if applicable.

If the step ends abnormally, any rows that SAS successfully processed after the last commit are rolled back and are therefore lost. Unless DBCOMMIT= 1, it is very likely that rows can be lost. The default value is 1000.

---

**Note:** A significant performance impact can result if you use this option from a SAS client session in SAS/SERVE or SAS/CONNECT environments to create or populate a newly created table. To prevent this, use the default value, ERRLIMIT=1.

---

### Specifics for Teradata

A rollback to the last checkpoint does not take place on reaching ERRLIMIT because the rows without errors have already been sent to Teradata.

SAS stops loading data when it reaches the specified number of errors and Fastload pauses. When Fastload pauses, you cannot use the table that is being loaded. Restart capability for Fastload is not yet supported, so you must manually delete the error tables before SAS can reload the table.

This option applies to TPT Load, Update, and Stream operations. For TPT, this option sets the value for TD\_ERROR\_LIMIT.

---

### Example: Specify the Number of Allowable Errors

In this example, SAS stops processing and issues a rollback to the DBMS at the occurrence of the 10th error. The MYDBLIB libref was assigned in a prior LIBNAME statement.

```
data mydblib.employee3 (errlimit=10);
  set mydblib.employees;
  where salary > 40000;
run;
```

---

## ESCAPE\_BACKSLASH= Data Set Option

Specifies whether backslashes in literals are preserved during data copy from a SAS data set to a table.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

|              |                                                  |
|--------------|--------------------------------------------------|
| Category:    | Data Set Control                                 |
| Default:     | NO                                               |
| Data source: | MySQL                                            |
| See:         | <a href="#">ESCAPE_BACKSLASH= LIBNAME option</a> |

## Syntax

**ESCAPE\_BACKSLASH=YES | NO**

### Syntax Description

#### YES

specifies that an additional backslash is inserted in every literal value that already contains a backslash.

#### NO

specifies that backslashes that exist in literal values are not preserved. An error results.

## Details

MySQL uses the backslash as an escape character. When data that is copied from a SAS data set to a MySQL table contains backslashes in literal values, the MySQL interface can preserve them if ESCAPE\_BACKSLASH=YES.

## Examples

### Example 1: Preserve Backslashes

In this example, SAS preserves the backslashes for x and y values.

```
libname out mysql user=myusr1 pw=mypwd1
      server=striper database=test port=3306;
data work.test;
  length x y z $10;
  x = "ABC";
  y = "DEF\";
  z = 'GHI\'';
run;
data out.test(escape_backslash=yes);
set work.test;
run;
```

The code successfully generates this INSERT statement.

```
INSERT INTO 'test' ('x','y','z')  VALUES ('ABC','DEF\\','GHI\\\\')
```

## Example 2: Use the Default Value (Do Not Preserve Backslashes)

For the prior example, here is the error that is displayed if ESCAPE\_BACKSLASH=NO.

```
ERROR: Execute error: You have an error in your SQL syntax;
check the manual that corresponds to your MySQL server version for the
right syntax to use near 'GHI\'\')' at line 1
```

## FASTEXPORT= Data Set Option

Specifies whether the SAS/ACCESS engine uses the TPT API to read data.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">FASTLOAD= data set option</a> , <a href="#">LOGDB= LIBNAME option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">QUERY_BAND= LIBNAME option</a> , <a href="#">QUERY_BAND= data set option</a> , <a href="#">SLEEP= LIBNAME option</a> , <a href="#">SLEEP= data set option</a> , <a href="#">TENACITY= LIBNAME option</a> , <a href="#">TENACITY= data set option</a> , <a href="#">TPT_BLOCK_SIZE= data set option</a> , <a href="#">TPT_DATA_ENCRYPTION= data set option</a> , <a href="#">TPT_LOG_TABLE= data set option</a> , <a href="#">TPT_MAX_SESSIONS= LIBNAME option</a> , <a href="#">TPT_MAX_SESSIONS= data set option</a> , <a href="#">TPT_MIN_SESSIONS= data set option</a> <a href="#">TPT_TRACE_LEVEL= data set option</a> , <a href="#">TPT_TRACE_LEVEL_INF= data set option</a> , <a href="#">TPT_UNICODE_PASSTHRU= LIBNAME option</a> |

## Syntax

**FASTEXPORT= YES | NO**

### Syntax Description

#### YES

specifies that data is loaded from Teradata into SAS using the TPT Export driver.

#### NO

specifies that the TPT Export driver is not to be used.

## Details

By using the TPT API, you can read data from a Teradata table without working directly with the stand-alone Teradata FastExport utility. When FASTEXPORT=YES, SAS uses the TPT API export driver for bulk reads. If SAS cannot use the TPT API—because of an error or because it is not installed on the system—SAS still tries to read the data. However, no error is produced. To see whether SAS used the TPT API to read data, look for this message in the SAS log:

NOTE: Teradata connection: TPT FastExport has read n row(s).

When you specify a query band on this option, specify the DBSLICEPARM=LIBNAME option. The query band is passed as a SESSION query band to the FastExport utility.

To see whether threaded reads are actually generated, turn on SAS tracing by specifying OPTIONS SASTRACE=",,d" in your program.

## Example

In this example, the TPT API reads SAS data from a Teradata table. Remember, SAS still tries to read data even if it cannot use the TPT API.

```
Libname tera Teradata user=myusrl pw=mypwd1 FASTEXPORT=YES;
/* Create data */
Data teratestdata;
Do i=1 to 100;
   Output;
End;
Run;
/* Read using FastExport TPT. This note appears in the SAS log if SAS uses TPT.
NOTE: Teradata connection: TPT FastExport has read
n row(s).*/
Data work.testdata;
Set teratestdata;
Run;
```

---

## FASTLOAD= Data Set Option

Specifies whether to use the TPT load driver.

|              |                                                                 |
|--------------|-----------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS) |
| Category:    | Bulk Loading                                                    |
| Default:     | NO                                                              |
| Data source: | Teradata                                                        |

See: BULKLOAD= LIBNAME option, BULKLOAD= data set option, DBSLICEPARM= LIBNAME option, DBSLICEPARM= data set option, DBSLICEPARM= system option  
 FASTLOAD= LIBNAME option , LOGDB= LIBNAME option, FASTEXPORT= LIBNAME option , Maximizing Teradata Load Performance, MULTILOAD= data set option, QUERY\_BAND= LIBNAME option, QUERY\_BAND= data set option, SLEEP= LIBNAME option, SLEEP= data set option, TENACITY= LIBNAME option, TENACITY= data set option, TPT= data set option, TPT= LIBNAME option, TPT\_DATA\_ENCRYPTION= data set option, TPT\_DATA\_ENCRYPTION= LIBNAME option, TPT\_UNICODE\_PASSTHRU= LIBNAME option

---

## Syntax

**FASTLOAD YES | NO**

### Syntax Description

#### YES

specifies that the SAS/ACCESS engine uses the TPT load driver to load the data.

#### NO

specifies that the SAS/ACCESS engine does not use the TPT load driver to load the data.

---

## Details

### Default Behavior for Data Transfer

By default for Teradata, you transfer data by using the Teradata Parallel Transporter (TPT) API. FastLoad is considered a legacy feature. To enable data transfer using the TPT API, enable the [TPT= LIBNAME option](#) or the [TPT= data set option](#). For more information, see [“Maximizing Teradata Load and Read Performance” on page 1337](#).

### Best Practice for Specifying FASTLOAD=

You can specify FASTLOAD= by using a data set option or LIBNAME option. The disadvantage of using the LIBNAME option is that, whenever you insert data from SAS into Teradata tables in the SAS library, the FastLoad protocol is used. This takes Teradata utility slots, which might also cause other load jobs to fail.

Specifying the FASTLOAD= data set option on a SAS library hides it from users. In a business intelligence environment, it might be difficult to determine whether this option is specified. If you are setting up SAS libraries that only ETL jobs use, it might be acceptable to use the LIBNAME option.

---

**Note:** A best practice recommendation is to use FASTLOAD= as a data set option unless you have a compelling reason to use it as a LIBNAME option.

---

```

libname mytera TERADATA server=teraserv user=bob pw=bob1;
/* Create and load a table using a Data step. SAS numeric is */
/* forced to be an INTEGER.   */
data mytera.table0 (FASTLOAD=YES DBTYPE= (x= INTEGER)) ;
  do x = 1 to 1000000;
    output;
  end; run;

/* Load an existing table using PROC APPEND. The table must */
/* meet certain requirements.                                     */
PROC SQL;
  CONNECT TO TERADATA (USER=bob PW=bob1 SERVER=teradata5500);
  EXECUTE (DROP TABLE loadThisTable) BY TERADATA;
  EXECUTE (COMMIT) BY TERADATA;
  EXECUTE (CREATE MULTISET TABLE loadThisTable ( a INTEGER , b
CHAR(10))
          PRIMARY INDEX (a)) BY TERADATA;
  EXECUTE (COMMIT) BY TERADATA;
QUIT;
DATA work.loadData;
  FORMAT b $10.;
  a = 1;
  output;
  b = 'One';
  output;
  a = 2;
  output;
  b = 'Two';
  output;
  a = 3;
  output;
  b = 'Three';
  output;
RUN;

libname mytera teradata server=teraserv user=bob pw=bob1;
PROC APPEND BASE=mytera.loadThisTable (FASTLOAD=YES
BL_LOG=BOB_APPEND_ERR)
  DATA=work.loadData;
RUN;

```

---

## FETCH\_IDENTITY= Data Set Option

Returns the value of the last inserted identity value.

|              |                                                                                              |
|--------------|----------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                     |
| Category:    | Data Set Control                                                                             |
| Default:     | NO                                                                                           |
| Data source: | DB2 under UNIX and PC Hosts                                                                  |
| See:         | <a href="#">FETCH_IDENTITY= LIBNAME option</a> , <a href="#">INSERTBUFF= data set option</a> |

---

## Syntax

**FETCH\_IDENTITY=YES | NO**

### Syntax Description

#### YES

returns the value of the last inserted identity value.

#### NO

disables this option.

---

## Details

You can use this option instead of issuing a separate SELECT statement after an INSERT statement. If FETCH\_IDENTITY=YES and the INSERT that is executed is a single-row INSERT, the engine calls the DB/2 identity\_val\_local() function and places the results into the SYSDB2\_LAST\_IDENTITY macro variable. Because the DB2 engine default is multirow inserts, you must set INSERTBUFF=1 to force a single-row INSERT.

---

## HDFS\_PRINCIPAL= Data Set Option

Specifies the Kerberos principal for HDFS.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS)

Category: Data Set Control

Default: none

Restriction: This option applies only when you configure HDFS to allow Kerberos authentication.

Requirement: To specify this option, you must first specify BULKLOAD=YES.

Data source: Impala

Note: Support for this data set option was added in SAS 9.4M2.

See: [BULKLOAD= LIBNAME option](#), [IMPALA\\_PRINCIPAL= LIBNAME option](#)

---

## Syntax

**HDFS\_PRINCIPAL='principal'**

## Required Argument

### *principal*

specifies the server's Kerberos service principal name (SPN). Surround the principal value with single or double quotation marks. For example, you might specify a principal value that is similar to the following code:

```
hdfs_principal='hdfs/hdfs_host.example.com@TEST.EXAMPLE.COM'
```

## IGNORE\_READ\_ONLY\_COLUMNS= Data Set Option

Specifies whether to ignore or include columns whose data types are read-only when generating an SQL statement for inserts or updates.

|              |                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                      |
| Category:    | Data Set Control                                                                                                                                                              |
| Alias:       | IGNORE_READONLY= [Greenplum, HAWQ, SAP IQ]                                                                                                                                    |
| Default:     | NO                                                                                                                                                                            |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Impala, Microsoft SQL Server, Netezza, ODBC, OLE DB, PostgreSQL, SAP HANA, SAP IQ, Vertica, Yellowbrick |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">IGNORE_READ_ONLY_COLUMNS=</a>                                                                                                                                     |

## Syntax

**IGNORE\_READ\_ONLY\_COLUMNS=**YES | NO

## Syntax Description

### **YES**

specifies that the SAS/ACCESS engine ignores columns whose data types are read-only when you are generating insert and update SQL statements

### **NO**

specifies that the SAS/ACCESS engine does not ignore columns whose data types are read-only when you are generating insert and update SQL statements

## Details

Several databases include data types that can be read-only, such as the Microsoft SQL Server timestamp data type. Several databases also have properties that allow certain data types to be read-only, such as the Microsoft SQL Server identity property.

When IGNORE\_READ\_ONLY\_COLUMNS=NO (the default) and a DBMS table contains a column that is read-only, an error is returned that the data could not be modified for that column.

## Example: Insert Data into a Table

For this example, a database that contains the table Products is created with two columns: ID and PRODUCT\_NAME. The ID column is specified by a read-only data type and PRODUCT\_NAME is a character column.

```
CREATE TABLE products (id int IDENTITY PRIMARY KEY, product_name varchar(40))
```

If you have a SAS data set that contains the name of your products, you can insert the data from the SAS data set into the Products table.

```
data work.products;
  id=1;
  product_name='screwdriver';
  output;
  id=2;
  product_name='hammer';
  output;
  id=3;
  product_name='saw';
  output;
  id=4;
  product_name='shovel';
  output;
run;
```

When IGNORE\_READ\_ONLY\_COLUMNS=NO (the default), the database returns an error because the ID column cannot be updated. However, if you set the option to YES and execute a PROC APPEND, the append succeeds and the generated SQL statement does not contain the ID column.

```
libname x odbc uid=myusr1 pwd=mypwd1 dsn=lupinss
  ignore_read_only_columns=yes;
options sastrace=',,d' sastraceloc=saslog nostsuffix;
proc append base=x.PRODUCTS data=work.products;
run;
```

---

## IN= Data Set Option

Lets you specify the database or tablespace in which you want to create a new table.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

|              |                                             |
|--------------|---------------------------------------------|
| Category:    | Data Set Control                            |
| Alias:       | CREATED_IN= [DB2 under UNIX and PC Hosts]   |
| Default:     | LIBNAME option value                        |
| Data source: | DB2 under UNIX and PC Hosts, DB2 under z/OS |
| See:         | <a href="#">IN= LIBNAME option</a>          |

## Syntax

**IN='database-name.tablespace-name' | database-name'**

### Syntax Description

***database-name.tablespace-name***

specifies the names of the database and tablespace, which are separated by a period.

***DATABASE database-name***

specifies only the database name. In this case, you specify the word DATABASE, a space, and the database name. Enclose the entire specification in single quotation marks.

## Details

The IN= option is relevant only when you are creating a new table. If you omit this option, the default is to create the table in the default database or tablespace.

## INSERT\_SQL= Data Set Option

Determines the method to use to insert rows into a data source.

|              |                                                                                          |
|--------------|------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                 |
| Category:    | Data Set Control                                                                         |
| Default:     | LIBNAME option value                                                                     |
| Data source: | Amazon Redshift, Microsoft SQL Server, Netezza, ODBC, OLE DB, PostgreSQL                 |
| See:         | <a href="#">INSERT_SQL= LIBNAME option</a> , <a href="#">INSERTBUFF= data set option</a> |

## Syntax

**INSERT\_SQL=YES | NO**

## Syntax Description

### YES

specifies that the SAS/ACCESS engine uses the data source's SQL insert method to insert new rows into a table.

### NO

specifies that the SAS/ACCESS engine uses an alternate (DBMS-specific) method to add new rows to a table.

## Details

Flat-file databases such as dBase, FoxPro, and text files have generally improved insert performance when `INSERT_SQL=NO`. Other databases might have inferior insert performance or might fail with this value. Therefore, you should experiment to determine the optimal value for your situation.

*Microsoft SQL Server:* The Microsoft SQL Server default is YES. When `INSERT_SQL=NO`, the `SQLSetPos (SQL_ADD)` function inserts rows in groups that are the size of the `INSERTBUFF=` value. The `SQLSetPos (SQL_ADD)` function does not work unless your ODBC driver supports it.

*ODBC:* The default for ODBC is YES, except for Microsoft Access, which has a default of NO. When `INSERT_SQL=NO`, the `SQLSetPos (SQL_ADD)` function inserts rows in groups that are the size of the `INSERTBUFF=` option value. The `SQLSetPos (SQL_ADD)` function does not work unless your ODBC driver supports it.

*OLE DB:* By default, the OLE DB interface attempts to use the most efficient row insertion method for each data source. You can use the `INSERT_SQL` option to override the default in the event that it is not optimal for your situation. The OLE DB alternate method (used when this option is set to NO) uses the OLE DB `IRowsetChange` interface.

## INSERTBUFF= Data Set Option

Specifies the number of rows in a single DBMS insert.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: LIBNAME option value

Restrictions: SAS allows the maximum number of rows that the DBMS allows, up to 32,767 rows.

When you insert rows with the `VIEWTABLE` window or the `FSVIEW` or `FSEDIT` procedure, use `INSERTBUFF=1` to prevent the DBMS interface from trying to insert multiple rows. These features do not support inserting more than one row at a time.

Additional driver-specific restrictions might apply.

The optimal value for this option varies with factors such as network type and available memory.

*DB2 under UNIX and PC Hosts:* If one row in the insert buffer fails, all rows in the insert buffer fail.

*MySQL:* Values greater than 0 activate the INSERTBUFF= option, and the engine calculates how many rows it can insert at one time, based on row size. If one row in the insert buffer fails, all rows in the insert buffer might fail, depending on your storage type.

Interactions: If you specify DBCOMMIT= with a value that is less than the value of INSERTBUFF=, then DBCOMMIT= overrides INSERTBUFF=.

*DB2 under UNIX and PC Hosts, Greenplum:* To use this option, you must specify INSERT\_SQL=YES.

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Greenplum, HAWQ, Impala, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for JDBC was added in SAS Viya 3.4.

Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Yellowbrick was added in SAS 9.4M7.

Tip: You might need to experiment with different values determine the best value for your site.

See: [DBCOMMIT= LIBNAME option](#), [DBCOMMIT= data set option](#), [INSERTBUFF= LIBNAME option](#), [INSERT\\_SQL LIBNAME option](#), [INSERT\\_SQL data set option](#), [READBUFF= LIBNAME option](#), [READBUFF= data set option](#)

## Syntax

**INSERTBUFF=***positive-integer*

### Syntax Description

#### *positive-integer*

specifies the number of rows to insert. SAS allows the maximum that the DBMS allows.

## KEYSET\_SIZE= Data Set Option

Specifies the number of rows in the cursor that the keyset drives.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Alias: KEYSET= [Greenplum, HAWQ, Microsoft SQL Server]

Default: LIBNAME option value

Requirement: This option is valid only when CURSOR\_TYPE=KEYSET\_DRIVEN.

Data source: Amazon Redshift, Greenplum, HAWQ, Microsoft SQL Server, ODBC, PostgreSQL

- Note: Support for Amazon Redshift, Greenplum, HAWQ, and PostgreSQL was added in SAS Viya 3.4.
- See: [KEYSET\\_SIZE= LIBNAME option](#)

## Syntax

**KEYSET\_SIZE=***number-of-rows*

### Syntax Description

#### ***number-of-rows***

specifies a positive integer from 0 through the number of rows in the cursor.

## Details

If KEYSET\_SIZE=0, the entire cursor is keyset driven.

If you specify a value greater than 0, that value indicates the number of rows within the cursor that function as a keyset-driven cursor. When you scroll beyond the bounds that KEYSET\_SIZE= specifies, the cursor becomes dynamic and new rows might be included in the cursor. This results in a new keyset, where the cursor functions as a keyset-driven cursor again.

When you specify a value between 1 and the number of rows in the cursor, the cursor is considered to be a mixed cursor. Part of the cursor functions as a keyset-driven cursor, and another part of the cursor functions as a dynamic cursor.

## LOCATION= Data Set Option

Lets you specify exactly where data resides.

- Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)
- Category: Data Access
- Alias: LOC=
- Default: LIBNAME option value
- Requirement: If you specify LOCATION=, you must also specify the AUTHID= data set option.
- Data source: DB2 under z/OS
- See: [AUTHID= data set option, LOCATION= LIBNAME option](#)

## Syntax

**LOCATION=***location-name*

---

## Details

The location name maps to the location in the SYSIBM.LOCATIONS catalog in the communication database.

In SAS/ACCESS Interface to DB2 under z/OS, the location is converted to the first level of a three-level table name: *location-name.AUTHID.TABLE*. The DB2 Distributed Data Facility (DDF) makes the connection implicitly to the remote DB2 subsystem when DB2 receives a three-level name in an SQL statement.

---

## LOCKTABLE= Data Set Option

Places exclusive or shared locks on tables.

|              |                                                                                             |
|--------------|---------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                    |
| Category:    | Data Set Control                                                                            |
| Default:     | LIBNAME option value                                                                        |
| Restriction: | If you omit LOCKTABLE=, no locking occurs.                                                  |
| Requirement: | You can lock tables only if you are the owner or have been granted the necessary privilege. |
| Data source: | Informix                                                                                    |
| See:         | <a href="#">LOCKTABLE= LIBNAME option</a>                                                   |

---

## Syntax

**LOCKTABLE=EXCLUSIVE | SHARE**

### Syntax Description

#### **EXCLUSIVE**

locks a table exclusively, preventing other users from accessing any table that you open in the libref.

#### **SHARE**

locks a table in shared mode. It allows other users or processes to read data from the tables but preventing users from updating data.

---

## MBUFSIZE= Data Set Option

Specifies the size of the shared memory buffers to use for transferring data from SAS to Teradata.

|           |                                                                                            |
|-----------|--------------------------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software) |
|-----------|--------------------------------------------------------------------------------------------|

|              |                                                                                                                                                                                 |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Category:    | Data Set Control                                                                                                                                                                |
| Default:     | 64K                                                                                                                                                                             |
| Requirement: | To specify this option, you must first specify MULTILOAD=YES.                                                                                                                   |
| Data source: | Teradata                                                                                                                                                                        |
| Tip:         | Use BUFFERS= to vary the number of buffers for data transfer from 1 to 8. Use MBUFSIZE= to vary the size of the shared memory buffers from the size of each data row up to 1MB. |
| See:         | <a href="#">BUFFERS= data set option</a> , <a href="#">MULTILOAD= data set option</a> , Teradata MultiLoad documentation, <a href="#">Using MultiLoad</a>                       |

## Syntax

**MBUFSIZE=***size-of-shared-memory-buffers*

### Syntax Description

#### ***size-of-shared-memory-buffers***

specifies a numeric value (between the size of a row being loaded and 1MB) that specifies the buffer size.

## Details

Two data set options are available for tuning the number and size of data buffers used for transferring data from SAS to Teradata.

When you use MULTILOAD=, data transfers from SAS to Teradata using shared memory segments. The default shared memory buffer size is 64K. The default number of shared memory buffers that are used for the transfer is 2.

## ML\_CHECKPOINT= Data Set Option

Specifies the interval between checkpoint operations in minutes.

|              |                                                                                                                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                            |
| Category:    | Data Set Control                                                                                                                                                                                      |
| Default:     | none (see "Details")                                                                                                                                                                                  |
| Requirement: | To specify this option, you must first specify MULTILOAD=YES.                                                                                                                                         |
| Data source: | Teradata                                                                                                                                                                                              |
| See:         | <a href="#">DBCOMMIT= LIBNAME option</a> , <a href="#">DBCOMMIT= data set option</a> , <a href="#">MULTILOAD= data set option</a> , Teradata MultiLoad documentation, <a href="#">Using MultiLoad</a> |

## Syntax

**ML\_CHECKPOINT=***checkpoint-rate*

### Syntax Description

#### ***checkpoint-rate***

a numeric value that specifies the interval between checkpoint operations in minutes.

---

## Details

If you do not specify a value for ML\_CHECKPOINT=, the Teradata Multiload default of 15 applies. If ML\_CHECKPOINT= is between 1 and 59 inclusive, checkpoints are taken at the specified intervals, in minutes. If ML\_CHECKPOINT= is greater than or equal to 60, a checkpoint occurs after a multiple of the specified rows is loaded.

ML\_CHECKPOINT= functions very similarly to CHECKPOINT in the native Teradata MultiLoad utility. However, it differs from DBCommit=. Both DBCommit= and its alias, CHECKPOINT, are disabled to prevent any conflict.

---

## ML\_ERROR1= Data Set Option

Specifies the name of a temporary table that MultiLoad uses to track errors that were generated during the acquisition phase of bulk -loading.

|               |                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                                                                                                                                                                                                 |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                                           |
| Default:      | none                                                                                                                                                                                                                                                                                                                                                                       |
| Restriction:  | Do not use ML_ERROR1 with the ML_LOG= data set option. ML_LOG= provides a common prefix for all temporary tables that the Teradata MultiLoad utility uses.                                                                                                                                                                                                                 |
| Requirements: | To specify this option, you must first specify MULTILOAD=YES.<br>When you restart a failed MultiLoad job, you must specify the same acquisition table from the earlier run so that the MultiLoad job can restart correctly. Using ML_RESTART=,ML_ERROR2=, and ML_WORK=, you must also specify the same log table, application error table, and work table upon restarting. |
| Data source:  | Teradata                                                                                                                                                                                                                                                                                                                                                                   |
| See:          | <a href="#">ML_ERROR2= data set option</a> , <a href="#">ML_LOG= data set option</a> , <a href="#">ML_RESTART= data set option</a> , <a href="#">ML_WORK= data set option</a> , <a href="#">MULTILOAD= data set option</a> , Teradata MultiLoad documentation, <a href="#">Using MultiLoad</a>                                                                             |

## Syntax

**ML\_ERROR1=***temporary-table-name*

### Syntax Description

***temporary-table-name***

specifies the name of a temporary table that MultiLoad uses to track errors that were generated during the acquisition phase of bulk loading.

### Details

Use this option to specify the name of a table to use for tracking errors that were generated during the acquisition phase of MultiLoad bulk loading. By default, the acquisition error table is named SAS\_ML\_ET\_<random>, where <random> is a random number.

---

## ML\_ERROR2= Data Set Option

Specifies the name of a temporary table that MultiLoad uses to track errors that were generated during the application phase of bulk loading.

|               |                                                                                                                                                                                                                                                                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                                                                                                                                                                                                  |
| Category:     | Data Set Control                                                                                                                                                                                                                                                                                                                                                            |
| Default:      | none                                                                                                                                                                                                                                                                                                                                                                        |
| Restriction:  | Do not use ML_ERROR2 with ML_LOG=, which provides a common prefix for all temporary tables that the Teradata MultiLoad utility uses.                                                                                                                                                                                                                                        |
| Requirements: | To specify this option, you must first specify MULTILOAD=YES.<br>When you restart a failed MultiLoad job, you must specify the same application table from the earlier run so that the MultiLoad job can restart correctly. Using ML_RESTART=, ML_ERROR1=, and ML_WORK=, you must also specify the same log table, acquisition error table, and work table upon restarting. |
| Data source:  | Teradata                                                                                                                                                                                                                                                                                                                                                                    |
| See:          | <a href="#">ML_ERROR1= data set option</a> , <a href="#">ML_LOG= data set option</a> , <a href="#">ML_RESTART= data set option</a> , <a href="#">ML_WORK= data set option</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">Teradata MultiLoad documentation</a> , <a href="#">Using MultiLoad</a>                                                             |

---

## Syntax

**ML\_ERROR2=***temporary-table-name*

## Syntax Description

***temporary-table-name***

specifies the name of a temporary table that MultiLoad uses to track errors that were generated during the application phase of bulk loading.

---

## Details

By default, the application error table is named SAS\_ML\_UT\_<random>, where <random> is a random number.

---

## ML\_LOG= Data Set Option

Specifies a prefix for the names of the temporary tables that MultiLoad uses during a bulk-loading operation.

|              |                                                                                                                                                                                                                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                                                                                                                                                                      |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                |
| Default:     | none                                                                                                                                                                                                                                                                                                                                            |
| Restriction: | Do not use ML_LOG= with ML_RESTART=,ML_ERROR1=,ML_ERROR2=, or ML_WORK= because ML_LOG= provide specific names to the temporary files.                                                                                                                                                                                                           |
| Requirement: | To specify this option, you must first specify MULTILOAD=YES.                                                                                                                                                                                                                                                                                   |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                        |
| See:         | <a href="#">ML_ERROR1= data set option</a> , <a href="#">ML_ERROR2= data set option</a> , <a href="#">ML_RESTART= data set option</a> , <a href="#">ML_RESTART= data set option</a> , <a href="#">ML_WORK= data set option</a> , <a href="#">MULTILOAD= data set option</a> , Teradata MultiLoad documentation, <a href="#">Using MultiLoad</a> |

---

## Syntax

***ML\_LOG=prefix-for-MultiLoad-temporary-tables***

## Syntax Description

***prefix-for-MultiLoad-temporary-tables***

specifies the prefix to use when naming Teradata tables that the Teradata MultiLoad utility uses during a bulk-loading operation.

## Details

The MultiLoad utility uses a log table, two error tables, and a work table while loading data to the target table. By default, here are the names for these tables, where *randnum* is a random number.

*Table 13.6 MultiLoad Temporary Tables*

| Temporary Table         | Table Name                |
|-------------------------|---------------------------|
| Restart table           | SAS_ML_RS_ <i>randnum</i> |
| Acquisition error table | SAS_ML_ET_ <i>randnum</i> |
| Application error table | SAS_ML_UT_ <i>randnum</i> |
| Work table              | SAS_ML_WT_ <i>randnum</i> |

To override the default names, here are the table names that would be generated if `ML_LOG=MY_LOAD`, for example.

*Table 13.7 Generated Tables Names*

| Temporary Table         | Table Name |
|-------------------------|------------|
| Restart table           | MY_LOAD_RS |
| Acquisition error table | MY_LOAD_ET |
| Application error table | MY_LOAD_UT |
| Work table              | MY_LOAD_WT |

SAS/ACCESS automatically deletes the error tables if no errors are logged. If there are errors, the tables are retained, and SAS/ACCESS issues a warning message that includes the names of the tables in error.

---

## ML\_RESTART= Data Set Option

Specifies the name of a temporary table that MultiLoad uses to track checkpoint information.

- Valid in: DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)
- Category: Data Set Control
- Default: none
- Restriction: Do not use `ML_RESTART=` with `ML_LOG=`, which provides a common prefix for all temporary tables that the Teradata MultiLoad utility uses.

|               |                                                                                                                                                                                                                                                                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Requirements: | To specify this option, you must first specify MULTILOAD=YES.<br>When you restart a failed MultiLoad job, you must specify the same application table from the earlier run so that the MultiLoad job can restart correctly. Using ML_RESTART=, ML_ERROR1=, and ML_WORK=, you must also specify the same log table, acquisition error table, and work table upon restarting. |
| Data source:  | Teradata                                                                                                                                                                                                                                                                                                                                                                    |
| See:          | <a href="#">ML_ERROR1= data set option</a> , <a href="#">ML_ERROR2 data set option</a> , <a href="#">ML_LOG= data set option</a> , <a href="#">ML_WORK= data set option</a> , <a href="#">MULTILOAD= data set option</a> , Teradata MultiLoad documentation, <a href="#">Using MultiLoad</a>                                                                                |

## Syntax

**ML\_RESTART=***temporary-table-name*

### Syntax Description

***temporary-table-name***

specifies the name of the temporary table that the Teradata [MultiLoad](#) utility uses to track checkpoint information.

## Details

Use this option to specify the name of a table to store checkpoint information. Upon restart, ML\_RESTART= is used to specify the name of the log table that you used for tracking checkpoint information in the earlier failed run.

## ML\_WORK= Data Set Option

Specifies the name of a temporary table that MultiLoad uses to store intermediate data.

|              |                                                                                                                                                                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                                                                                                                      |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                |
| Default:     | none                                                                                                                                                                                                                                                                                            |
| Restriction: | Do not use ML_WORK= with ML_LOG=, which provides a common prefix for all temporary tables that the Teradata MultiLoad utility uses.                                                                                                                                                             |
| Requirement: | To specify this option, you must first specify MULTILOAD=YES.                                                                                                                                                                                                                                   |
| Data source: | Teradata                                                                                                                                                                                                                                                                                        |
| See:         | <a href="#">ML_ERROR1= data set option</a> , <a href="#">ML_ERROR2 data set option</a> , <a href="#">ML_LOG= data set option</a> , <a href="#">ML_RESTART= data set option</a> , <a href="#">MULTILOAD= data set option</a> , Teradata MultiLoad documentation, <a href="#">Using MultiLoad</a> |

## Syntax

**ML\_WORK=***temporary-table-name*

### Syntax Description

#### *temporary-table-name*

specifies the name of a temporary table that MultiLoad uses to store intermediate data that the [MultiLoad](#) utility receives during bulk loading.

## Details

Use this option to specify the name of the table to use for tracking intermediate data that the MultiLoad utility received during bulk loading. When you restart the job, use **ML\_WORK=** to specify the name of the table for tracking intermediate data during a previously failed MultiLoad job.

---

## MULTILOAD= Data Set Option

Specifies whether Teradata Insert and Append operations should use the Teradata MultiLoad utility and TPT update driver.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| See:         | <a href="#">LOGDB= LIBNAME option</a> , “Maximizing Teradata Load and Read Performance”, <a href="#">MULTILOAD= data set option</a> , <a href="#">MULTISTMT= data set option</a> , <a href="#">QUERY_BAND= data set option</a> , <a href="#">TPT= LIBNAME option</a> , <a href="#">TPT_APPL_PHASE= data set option</a> , <a href="#">TPT_BUFFER_SIZE= data set option</a> , <a href="#">TPT_CHECKPOINT_DATA= data set option</a> , <a href="#">TPT_DATA_ENCRYPTION= data set option</a> , <a href="#">TPT_ERROR_TABLE_1= data set option</a> , <a href="#">TPT_ERROR_TABLE_2= data set option</a> , <a href="#">TPT_LOG_TABLE= data set option</a> , <a href="#">TPT_MAX_SESSIONS= data set option</a> , <a href="#">TPT_MIN_SESSIONS= data set option</a> , <a href="#">TPT_RESTART= data set option</a> , <a href="#">TPT_TRACE_LEVEL= data set option</a> , <a href="#">TPT_TRACE_LEVEL_INF= data set option</a> , <a href="#">TPT_TRACE_OUTPUT= data set option</a> , <a href="#">TPT_UNICODE_PASSTHRU= LIBNAME option</a> , <a href="#">TPT_WORK_TABLE= data set option</a> |

---

## Syntax

**MULTILOAD=**[YES](#) | [NO](#)

## Syntax Description

### YES

uses the Teradata Parallel Transporter (TPT) API to load data, if available.

### NO

sends inserts to Teradata tables one row at a time.

## Details

### Bulk Loading

The SAS/ACCESS MultiLoad facility provides a bulk-loading method of loading both empty and existing Teradata tables. Unlike FastLoad, MultiLoad can append data to existing tables.

To determine whether threaded Reads are actually generated, turn on SAS tracing by specifying `OPTIONS SASTRACE=" , , ,d"`; in your program.

### Temporary Processing Tables

The Teradata MultiLoad utility uses four temporary processing tables when it performs the bulk-loading operation. It uses a log table to track restart information, two error tables to track errors, and a work table to hold data before the Insert operation is made.

By default, the SAS/ACCESS MultiLoad facility generates names for these temporary processing tables. To specify a different name for these tables, use `TPT_CHECKPOINT_DATA=`, `TPT_ERROR_TABLE_1=`, `TPT_ERROR_TABLE_2=`, and `TPT_WORK_TABLE=` data set options, respectively.

| Temporary Processing Table | Default Table Name     |
|----------------------------|------------------------|
| Restart table              | <i>target-table_RS</i> |
| Acquisition error table    | <i>target-table_ET</i> |
| Application error table    | <i>target-table_UV</i> |
| Work table                 | <i>target-table_WT</i> |

You can use `ML_LOG=` to specify a prefix for the temporary processing table names that MultiLoad uses.

Here is the order that is used for naming the temporary processing tables that MultiLoad uses:

- 1 If you specify `ML_LOG=`, the prefix that you specified is used when naming temporary processing tables for MultiLoad.

- 2 If you do not specify ML\_LOG=, the values that you specified for TPT\_CHECKPOINT\_DATA=, TPT\_ERROR\_TABLE\_1=, TPT\_ERROR\_TABLE\_2=, and TPT\_WORK\_TABLE= are used.
- 3 If you do not specify any table naming options, temporary processing table names are generated by default.

## Upsert Processing for TPT

The *upsert* feature for Teradata performs a combination of updates and inserts in one step. When updating a master table with a transaction table, an UPDATE statement is first issued on the master table using a row of data from the transaction table. If no target row exists to satisfy the update, an INSERT statement adds the transaction row to the master table. To use the upsert feature, specify [UPINSERT=YES](#).

Upsert processing requires a WHERE clause that refers to the primary index of the target table that identifies the rows to update. When UPINSERT=YES, Teradata builds a WHERE condition, by default, based on the primary index columns of the target table. To specify the WHERE clause for an upsert, use the [UPINSERT\\_CONDITION= data set option](#). To specify one or more columns that the WHERE clause applies to, use the [UPINSERT\\_WHERE= data set option](#).

## Examples

### Example 1: Load SAS Data to an Alternate Database

This example uses MultiLoad to load SAS data to an alternate database. It specifies database=mloaduser in the LIBNAME statement.

```
libname trlib teradata user=myusr1 pw=mypwd1
      server=dbc database=mloaduser;
/*MultiLoad 20000 observations into alternate database mloaduser */
data trlib.trmload14(DBCREATE_TABLE_OPTS="PRIMARY INDEX(IDNUM)"
      MultiLoad=yes ML_LOG=TRMLOAD14 ML_CHECKPOINT=5000);
  set permdata.BIG1MIL(drop=year obs=20000);
run;
```

### Example 2: Extract Data from One Table to Another

This example extracts data from one table using FastExport and loads data into another table using MultiLoad.

```
libname trlib teradata user=myusr1 pw=mypwd1 server=dbc;
/* Create data to load */
data trlib.trodd(DBCREATE_TABLE_OPTS="PRIMARY INDEX(IDNUM)" MultiLoad=yes);
  set permdata.BIG1MIL(drop=year obs=10000);
  where mod(IDNUM,2)=1;
run;
/* FastExport from one table and MultiLoad into another */
proc append data=trlib.treven(dbsliceparm=all)
```

```
base=trlib.trall(MultiLOAD=YES) ;
run;
```

## MULTISTMT= Data Set Option

Specifies whether insert statements are sent to Teradata one at a time or in a group.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Restriction: | You cannot currently use MULTISTMT= with the ERRLIMIT= option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| See:         | <a href="#">DBCOMMIT= LIBNAME option</a> , <a href="#">DBCOMMIT= data set option</a> , <a href="#">ERRLIMIT= LIBNAME option</a> , <a href="#">ERRLIMIT= data set option</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">MULTISTMT= LIBNAME option</a> <a href="#">QUERY_BAND= data set option</a> , <a href="#">TPT_APPL_PHASE= data set option</a> , <a href="#">TPT_BUFFER_SIZE= data set option</a> , <a href="#">TPT_CHECKPOINT_DATA= data set option</a> on page 623, <a href="#">TPT_DATA_ENCRYPTION= data set option</a> , <a href="#">TPT_ERROR_TABLE_1= data set option</a> , <a href="#">TPT_ERROR_TABLE_2= data set option</a> , <a href="#">TPT_LOG_TABLE= data set option</a> , <a href="#">TPT_MAX_SESSIONS= LIBNAME option</a> , <a href="#">TPT_MAX_SESSIONS= data set option</a> , <a href="#">TPT_MIN_SESSIONS= data set option</a> , <a href="#">TPT_PACK= data set option</a> , <a href="#">TPT_PACKMAXIMUM= data set option</a> , <a href="#">TPT_RESTART= data set option</a> , <a href="#">TPT_TRACE_LEVEL= data set option</a> , <a href="#">TPT_TRACE_LEVEL_INF= data set option</a> , <a href="#">TPT_TRACE_OUTPUT= data set option</a> , <a href="#">TPT_UNICODE_PASSTHRU= LIBNAME option</a> |

## Syntax

**MULTISTMT=YES | NO**

### Syntax Description

#### YES

use TPT stream processing and use the maximum buffer size available for the Teradata Client TTU version and the Teradata DBS version that is being accessed.

#### NO

send inserts to Teradata one row at a time.

## Details

When you request multi-statement inserts, SAS first determines how many insert statements it can send to Teradata. Several factors determine the actual number of statements that SAS can send—for example:

- how many SQL insert statements can fit in the available buffer
- how many data rows can fit in the available buffer
- how many inserts the Teradata server chooses to accept

When you need to insert large volumes of data, you can significantly improve performance by using MULTISTMT= instead of inserting only single-row.

When you also specify DBCommit=, SAS determines the number of insert statements to send together at one time. It uses the smaller of the DBCommit= value and the number of insert statements that can fit in a buffer.

The temporary processing tables that are accessed by the TPT Multistmt utility are *target-table\_ET* and *target-table\_RS*. To specify a different table name, use the TPT\_ERROR\_TABLE\_1= and TPT\_CHECKPOINT\_DATA= data set options, respectively.

## Examples

### Example 1: Send and Insert Statements One at a Time

This example shows how to send insert statements one at a time to Teradata.

```
libname user teradata user=myusr1 pw=XXXXXX server=dbc;
proc datasets library=user;
  delete testdata;run;
data usertestdata(DBTYPE=(I="INT") MULTISTMT=NO);
  do i=1 to 50;
    output;
  end;
run;
```

### Example 2: Send 100 Rows at a Time

In this example, DBCommit=100. Therefore, SAS issues a commit after every 100 rows, sending only 100 rows at a time.

```
libname user teradata user=myusr1 pw=XXXXXX server=dbc;

proc datasets library=user;
  delete testdata;run;

data usertestdata(MULTISTMT=YES DBCOMM=100);
  do i=1 to 1000;
    output;
  end;
run;
```

## Example 3: Send a Specified Group of Rows at a Time

In this example, DBCommit=1000, which is much higher than in the previous example. SAS sends as many rows as it can fit in the buffer at a time (up to 1000), and it issues a commit after every 1000 rows. If only 600 can fit, 600 are sent to the database. It is followed by the remaining 400—the difference between 1000 and the initial 600 that were already sent. SAS then commits all rows.

```
libname user teradata user=myusr1 pw=XXXXXX server=dbc;

proc datasets library=user;
  delete testdata;run;

data usertestdata(MULTISTMT=YES DBCommit=1000);
do i=1 to 10000;
  output;
  end;
run;
```

## Example 4: Use a Global Options to Store a Temporary Processing Table

This example specifies CONNECTION=GLOBAL for all tables, creates a global temporary processing table, and stores the table in the current database schema.

```
libname user teradata user=myusr1 pw=XXXXXX server=dbc connection=global;
proc datasets library=user;
  delete temp1;run;

proc sql;
  connect to teradata(user=myusr1 pw=XXXXXXXX server=dbc connection=global);
  execute (CREATE GLOBAL TEMPORARY TABLE temp1 (col1 INT )
    ON COMMIT PRESERVE ROWS) by teradata;
  execute (COMMIT WORK) by teradata;
quit;
data work.test;
  do col1=1 to 1000;
    output;
  end;
run;
proc append data=work.test base=user.temp1(multistmt=yes);
run;
```

## NULLCHAR= Data Set Option

Indicates how missing SAS character values are handled during insert, update, DBINDEX=, and DBKEY= processing.

|           |                                                                          |
|-----------|--------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category: | Data Set Control                                                         |

|              |                                                                                                                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Default:     | SAS                                                                                                                                                                                                                                                                                               |
| Restriction: | The NULLCHAR= data set option does not apply to binary data types, such as the SQL_LONGVARBINARY type in ODBC or the VARBINARY type in Netezza.                                                                                                                                                   |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                             |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">DBINDEX= data set option</a> , <a href="#">DBKEY= data set option</a> , <a href="#">DBNULL= data set option</a> , <a href="#">NULLCHARVAL= data set option</a>                                                                            |

## Syntax

**NULLCHAR=SAS | YES | NO**

### Syntax Description

#### SAS

indicates that missing character values in SAS data sets are treated as NULL values if the DBMS allows these. Otherwise, they are treated as the NULLCHARVAL= value.

#### YES

indicates that missing character values in SAS data sets are treated as NULL values if the DBMS allows these. Otherwise, an error is returned.

#### NO

indicates that missing character values in SAS data sets are treated as the NULLCHARVAL= value—regardless of whether the DBMS allows NULL values for the column.

## Details

This option affects insert and update processing. It also applies when you use DBINDEX= and DBKEY=.

It works with NULLCHARVAL=, which determines what is inserted when NULL values are not allowed. The DBMS treats all missing SAS numeric values (represented in SAS as '.') as NULLvalues.

*Oracle*: For interactions between NULLCHAR= and BULKLOAD=ZX`11, see the [bulk-load topic](#) in the Oracle section.

---

## NULLCHARVAL= Data Set Option

Specifies the character string that replaces missing SAS character values during insert, update, DBINDEX=, and DBKEY= processing.

|              |                                                                                                                                                                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                          |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                  |
| Default:     | a blank character                                                                                                                                                                                                                                                                                 |
| Restriction: | The NULLCHARVAL= data set option does not apply to binary data types, such as the SQL_LONGVARBINARY type in ODBC or the VARBINARY type in Netezza.                                                                                                                                                |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                             |
| Notes:       | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> |
| See:         | <a href="#">DBFORCE= data set option</a> , <a href="#">DBINDEX= data set option</a> , <a href="#">DBKEY= data set option</a> , <a href="#">DBNULL= data set option</a> , <a href="#">NULLCHAR= data set option</a>                                                                                |

---

## Syntax

**NULLCHARVAL='character-string'**

---

## Details

This option affects insert and update processing and also applies when you use DBINDEX= and DBKEY=.

It also works with NULLCHAR= to determine whether a missing SAS character value is treated as a NULL value. If NULLCHARVAL= is longer than the maximum column width, the string is truncated if DBFORCE=YES or the operation fails if DBFORCE=NO.

---

## OR\_IDENTITY\_COLS= Data Set Option

specifies columns to be used to simulate an identity column.

|           |                                                                          |
|-----------|--------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
|-----------|--------------------------------------------------------------------------|

|              |                                                                                        |
|--------------|----------------------------------------------------------------------------------------|
| Category:    | Data Set Control                                                                       |
| Default:     | none                                                                                   |
| Interaction: | The format of the user sequence for this option depends on the value of BULKLOAD.      |
| Data source: | Oracle                                                                                 |
| Note:        | Support for this data set option was added for SAS 9.4.                                |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">, BULKLOAD= LIBNAME option</a> |

## Syntax

**OR\_IDENTITY\_COLS=(*column-name-1='user-sequence-1'*  
*<column-name-2='user-sequence-2' ... column-name-n='user-sequence-n'>*)**

### Syntax Description

#### ***column-name-N***

specifies the name of an identity column.

#### ***user-sequence-N***

specifies the expression that is used to generate the identity column values.

When BULKLOAD=YES, the user sequence should take the following form:

`sequence(max, n)`

When BULKLOAD=NO, the user sequence should take the following form:

`<user-created-sequence>.nextval`

## Details

When BULKLOAD=YES, you provide the name of an identity column and the expression that is used to generate identity values. You provide the expression in the form `sequence(max, n)`. This expression says that the values in the identity column are generated by taking the maximum identity value and adding the value *n*. For example, the expression `sequence(max, 2)` increments generated identity values by 2.

When BULKLOAD=NO, you provide the name of an identity column and the expression that identifies the identity values. You provide the expression in the form `user-created-sequence.nextval`, such as `t1_id.nextval`.

## OR\_PARTITION= Data Set Option

Allows reading, updating, and deleting from a particular partition in a partitioned table, also inserting and bulk loading into a particular partition in a partitioned table.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

|              |                                                                                                                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Category:    | Data Set Control                                                                                                                                                                                        |
| Default:     | none                                                                                                                                                                                                    |
| Restriction: | The partition name must be valid or an error occurs.                                                                                                                                                    |
| Data source: | Oracle                                                                                                                                                                                                  |
| Tip:         | This option is appropriate when reading, updating, and deleting from a partitioned table, also when inserting into a partitioned table or bulk loading to a table. You can use it to boost performance. |

## Syntax

**OR\_PARTITION=***name of a partition in a partitioned Oracle table*

### Syntax Description

**name of a partition in a partitioned Oracle table**  
specifies the partition name.

## Details

Use this option in cases where you are working with only one particular partition at a time in a partitioned table. Specifying this option boosts performance because you are limiting your access to only one partition of a table instead of the entire table.

## Examples

### Example 1: Read, Update, Delete, Load, and Insert from a Partitioned Table

This example shows one way that you can use this option.

```
libname x oracle user=myusr1 pw=mypwd1 path=mypath;
proc datasets library=x;
  delete orparttest;run;
data x.ORparttest ( dbtpe=(NUM='int')
  DBCREATE_TABLE_OPTS='partition by range (NUM)
    (partition p1 values less than (11),
     partition p2 values less than (21),
     partition p3 values less than (31),
     partition p4 values less than (41),
     partition p5 values less than (51),
     partition p6 values less than (61),
     partition p7 values less than (71),
     partition p8 values less than (81)
    )' );
```

```

do i=1 to 80;
  NUM=i;
  output;
end;
run;
options sastrace=",t,d" sastraceloc=saslog nostsuffix;
/* input */
proc print data=x.orparttest ( or_partition=p4 );
run;
/* update */
proc sql;
/* update should fail with 14402, 00000, "updating partition key column
   would cause a partition change"
// *Cause: An UPDATE statement attempted to change the value of a
//         partition key column causing migration of the row to
//         another partition.
// *Action: Do not attempt to update a partition key column or make
//         sure that the new partition key is within the range
//         containing the old partition key.
*/
update x.orparttest ( or_partition=p4 ) set num=100;
update x.orparttest ( or_partition=p4 ) set num=35;
select * from x.orparttest ( or_partition=p4 );
select * from x.orparttest ( or_partition=p8 );
/* delete */
delete from x.orparttest ( or_partition=p4 );
select * from x.orparttest;
quit;
/* load to an existing table */
data new; do i=31 to 39; num=i; output;end;
run;
data new2; do i=1 to 9; num=i; output;end;
run;
proc append base= x.orparttest ( or_partition=p4 ) data= new;
run;
/* Insert should fail 14401, 00000, "inserted partition key
   is outside specified partition"
// *Cause: The concatenated partition key of an inserted record is
//         outside the ranges of the two concatenated partition bound
//         lists that delimit the partition named in the INSERT statement.
// *Action: Do not insert the key or insert it in another partition.
*/
proc append base= x.orparttest ( or_partition=p4 ) data= new2;
run;
/* load to an existing table */
proc append base= x.orparttest ( or_partition=p4 bulkload=yes
bl_load_method=truncate ) data= new;
run;
/* insert should fail 14401 */
proc append base= x.orparttest ( or_partition=p4 bulkload=yes
bl_load_method=truncate ) data= new2;
run;

```

## Example 2: Create and Manipulate a Partitioned Table

Here are a series of sample scenarios that illustrate how you can use this option. The first shows how to create the ORPARTTEST table, on which all remaining examples depend.

```
libname x oracle user=myusr1 pw=mypwd1 path=mypath;
proc datasets library=x;
  delete orparttest;run;
data x.ORparttest ( dbtype=(NUM='int')
  DBCREATE_TABLE_OPTS='partition by range (NUM)
    (partition p1 values less than (11),
     partition p2 values less than (21),
     partition p3 values less than (31),
     partition p4 values less than (41),
     partition p5 values less than (51),
     partition p6 values less than (61),
     partition p7 values less than (71),
     partition p8 values less than (81)
    )' );
do i=1 to 80;
  NUM=i; output;
end;
run;
```

In this example, only the P4 partition is read.

```
proc print data=x.orparttest ( or_partition=p4 );
run;
```

Next, rows that belong to only the single P4 partition are updated.

```
proc sql;
update x.orparttest ( or_partition=p4 ) set num=35;
quit;
```

Although this code shows how a particular partition can be updated, updates and inserts to the partition key column can be done so that data must be migrated to a different partition in the table. This next example fails because the value 100 does not belong to the P4 partition.

```
proc sql;
update x.orparttest ( or_partition=p4 ) set num=100;
quit;
```

In this example, all rows in the P4 partition are deleted.

```
proc sql;
delete from x.orparttest ( or_partition=p4 );
quit;
```

Next, rows are added to the P4 partition in the table.

```
data new;
  do i=31 to 39; num=i; output;end;
run;
proc append base= x.orparttest ( or_partition=p4 );
  data= new;
run;
```

This example also adds rows to the P4 partition but uses the SQL\*Loader instead.

```
proc append base= x.orparttest ( or_partition=p4 bulkload=yes );
  data= new;
run;
```

## OR\_UPD\_NOWHERE= Data Set Option

Specifies whether SAS uses an extra WHERE clause when updating rows with no locking.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Categories: Data Access

Data Set Control

Alias: ORACLE\_73\_OR\_ABOVE=

Default: LIBNAME option value

Requirement: Due to the published Oracle bug 440366, sometimes an update on a row fails even if the row has not changed. Oracle offers this solution: When you create a table, increase the number of INITTRANS to at least 3 for the table.

Data source: Oracle

See: Locking in the Oracle Interface, OR\_UPD\_NOWHERE= LIBNAME option, SASTRACE= system option, SASTRACELOC= system option, UPDATE\_LOCK\_TYPE= data set option

## Syntax

**OR\_UPD\_NOWHERE=YES | NO**

### Syntax Description

#### YES

specifies that SAS does not use an additional WHERE clause to determine whether each row has changed since it was read. Instead, SAS uses the SERIALIZABLE isolation level (available with Oracle7.3 and later) for update locking. If a row changes after the serializable transaction starts, the update on that row fails.

#### NO

specifies that SAS uses an additional WHERE clause to determine whether each row has changed since it was read. If a row has changed since being read, the update fails.

## Details

Use this option when you are updating rows without locking ([UPDATE\\_LOCK\\_TYPE=NOLOCK](#)).

By default (OR\_UPD\_NOWHERE=YES), updates are performed in serializable transactions so that you can avoid problems with extra WHERE clause processing and potential WHERE clause floating-point precision.

---

## Example: Create and Update a Table

In this example, you create a small Oracle table, TEST. You then update it once by using the default value (OR\_UPD\_NOWHERE=YES) and once by specifying OR\_UPD\_NOWHERE=NO.

```
libname oralib oracle user=myusr1 pw=mypwd1 update_lock_type=no;
data oralib.test;
c1=1;
c2=2;
c3=3;
run;
options sastrace=",,d" sastraceloc=saslog;
proc sql;
update oralib.test set c2=22;
update oralib.test(or_upd_nowhere=no) set c2=222;
quit;
```

This code uses the SASTRACE= and SASTRACELOC= system options to send the output to the SAS log.

---

## ORHINTS= Data Set Option

Specifies Oracle hints to pass to Oracle from a SAS statement or SQL procedure.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Default:     | none                                                                     |
| Data source: | Oracle                                                                   |

---

## Syntax

**ORHINTS='Oracle-hint'**

### Syntax Description

***Oracle-hint***

specifies an Oracle hint for SAS/ACCESS to pass to the DBMS as part of an SQL query.

## Details

If you specify an Oracle hint, SAS passes the hint to Oracle. If you omit ORHINTS=, SAS does not send any hints to Oracle.

### Example: Pass a Hint

This example runs a SAS procedure on DBMS data and SAS converts the procedure to one or more SQL queries. ORHINTS= lets you specify an Oracle hint for SAS to pass as part of the SQL query.

```
libname mydblib oracle user=myusr1 password=mypwd1 path='myorapath';
proc print data=mydblib.payroll(orhints='/*+ ALL_ROWS */');
run;
```

In this example, SAS sends the Oracle hint '/\*+ ALL\_ROWS \*/' to Oracle as part of this statement:

```
SELECT /*+ ALL_ROWS */ * FROM PAYROLL
```

---

## OVERLAPS= Data Set Option

determines whether the values for BUS\_START and BUS\_END can overlap between active records for the specified columns.

|              |                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                     |
| Category:    | Data Set Control                                                                                                                                                                                             |
| Default:     | <i>column names separated by commas</i>                                                                                                                                                                      |
| Interaction: | allowed for the BUSINESS_TIME period                                                                                                                                                                         |
| Data source: | DB2 z/OS                                                                                                                                                                                                     |
| See:         | <a href="#">TEMPORAL= data set option</a> , <a href="#">BUSINESS_TIMEFRAME= data set option</a> ,<br><a href="#">SYSTEM_TIMEFRAME = data set option</a> , <a href="#">BUSINESS_DATATYPE= data set option</a> |

---

## Syntax

**OVERLAPS='*column names*'**

### Syntax Description

#### ***column names***

a list of column names, separated by commas, for which business time period overlap is not allowed.

## Details

Here is an example that uses the OVERLAPS= option:

```
proc sql;
  create table policy (dbnull=(_all_=no) temporal=business OVERLAPS='id,vin')
    (id,int,vin varchar(10));
  <additional statements>
quit;
```

---

## PARMDEFAULT= Data Set Option

Specifies whether the SAP HANA engine uses the defaults for variables and parameters that are specified in the metadata in SAP HANA.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Alias:       | PARMDEFAULT, USE_PARAMETER_DEFAULT                                       |
| Default:     | none                                                                     |
| Data source: | SAP HANA                                                                 |

---

## Syntax

**PARMDEFAULT=YES | NO**

### Syntax Description

#### **YES**

specifies to look up metadata for variables and parameters. This enables you to apply defaults for variables in a WHERE clause, and defaults for input parameters using the PLACEHOLDER syntax.

#### **NO**

specifies to not look up metadata for variables and parameters.

---

## Details

The default for the PARMDEFAULT= option is YES if the PARMSTRING= LIBNAME or data set option is specified. The default is NO if no PARMSTRING= option is specified.

Applying the defaults requires additional queries to the metadata. It can be switched off to avoid making unnecessary queries.

## PARMSTRING= Data Set Option

Specifies a quoted string of variable name and value pairs separated by a comma, or a placeholder string.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Alias:       | PARAMETERS                                                               |
| Default:     | none                                                                     |
| Data source: | SAP HANA                                                                 |

## Syntax

```
PARMSTRING=<"variable-name1=variable-value1,variable-name2=variable-value2,
...">
    <'"PLACEHOLDER' = ('variable-name1', 'variable-value1'),<'PLACEHOLDER'=
('variable-name2', 'variable-value2'),>...">
```

## Syntax Description

### "variable-name1=variable-value1"

specifies the variable name and value pair. More than one pair can be specified and must be separated by a comma.

### "'PLACEHOLDER' = ('variable-name1', 'variable-value1')"

specifies the variable name and value pair as a placeholder.

**Note:** You can also combine a name and value pair and a placeholder:

```
PARMSTRING="variable-name1=variable-value1",
<'PLACEHOLDER'= ('variable-name2', 'variable-value2')"
```

## Details

When you specify the variable name and value pairs, the SAS/ACCESS engine locates the variable input parameter in the SAP HANA metadata. The value is applied either as a WHERE clause for variables, or it generates and applies a PLACEHOLDER= string for passing the input parameters to the view execution. If the variable input parameter is not found in metadata, it is appended as part of a PLACEHOLDER= string to the generated SQL string.

If the user specifies a placeholder string, the string is passed directly to the SAP HANA query to be processed in SAP HANA.

Here are some syntax examples:

```

PARMSTRING = "parm_price=30"

PARMSTRING = "'PLACEHOLDER' = ('$$parm_product$$', 'Tablet')"

PARMSTRING = "PLACEHOLDER. $$parm_category$$ =>'Notebooks'"

```

When a PARMSTRING= LIBNAME option and a PARMSTRING= data set option are both specified, the PLACEHOLDER= string becomes a combination of both of the parameters and is passed to SAP HANA. An input parameter can occur only once as a placeholder in the SQL statement. If a parameter appears in the LIBNAME option and the data set option, the SAS/ACCESS engine tries to resolve this by passing only a fragment from the data set option.

You can access a view using its synonym.

Here is an example of the PARMSTRING= data set option:

```

data work.a;
  set a.PRODUCT_LIST_WITH_PARAM_DEF_S
    (parmstring="parm_category='Notebooks'");
run;

```

## Comparisons

The PARMSTRING= data set option is applied to the table or view that is specified.

The PARMSTRING= LIBNAME option is applied to all column tables and column views in the library. It will not be applied to row store objects.

## PARTITION\_KEY= Data Set Option

Specifies the column name to use as the partition key for creating fact tables.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: none

Requirements: To create a data set in Aster without error, you must either specify DIMENSION=YES (LIBNAME or data set option) or specify a partition key in the PARTITION\_KEY= (LIBNAME or data set) option.

You must enclose the column name in quotation marks.

Data source: Aster

See: [DIMENSION= LIBNAME option](#), [DIMENSION= data set option](#), [PARTITION\\_KEY= LIBNAME option](#) [contains examples]

## Syntax

**PARTITION\_KEY='column-name'**

## Details

Aster uses two table types, dimension and fact tables.

---

## PARTITIONED\_BY= Data Set Option

Specifies the column name to use as the partition key for creating fact tables.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Aliases: PARTITION\_BY

PARTITION

Default: none

Data source: Impala

Note: Support for this data set option was added in SAS 9.4M2.

---

## Syntax

**PARTITIONED\_BY=(column-name1 <... column-nameN>)**

### Required Argument

**column-name1 <... column-nameN>**

specifies the column names that are used to partition the table that is being bulk loaded. Separate multiple values with spaces and include the list in parentheses.

---

## Example: Load a Partitioned Table

This example shows how to load a table that is partitioned by columns Col4 and Col5.

```
options sastrace=',,,d' sastraceloc=saslog
      nostsuffix msglevel=i;
data x.partitionsample(partitioned_by=(col4 col5) dbtype=(col4='int' col5='int') );
  col1=1;
  col2='This is a test';
  col3='*****0*****';
  col4=1;
  col5=10;
run;
```

```

IMPALA_2: Executed: on connection 2
CREATE TABLE `default`.`partitionsample` (`col1` double,`col2` VARCHAR(14),
`col3` VARCHAR(19)) PARTITIONED BY (`col4` int,`col5` int)

IMPALA_3: Prepared: on connection 2
SELECT * FROM `default`.`partitionsample` 

IMPALA_4: Prepared: on connection 2
INSERT INTO `default`.`partitionsample` (`col1`, `col2`, `col3`, `col4`,
`col5`) VALUES ( ? , ? , ? , ? , ? )

IMPALA_5: Executed: on connection 2
Prepared statement IMPALA_4

```

---

## PG\_IDENTITY\_COLS= Data Set Option

Specifies that sequencing is being applied to one or more columns.

|               |                                                                                                                                                                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                    |
| Category:     | Data Set Control                                                                                                                                                                                                                                                            |
| Default:      | none                                                                                                                                                                                                                                                                        |
| Restrictions: | <p>This option is not supported when INSERT_SQL=NO or with any operation that modifies the cursor position in a table.</p> <p>When BULKLOAD=YES, the sequence name must be associated with a Sequence object that was created previously by using the APPEND procedure.</p> |
| Data source:  | PostgreSQL, Yellowbrick                                                                                                                                                                                                                                                     |
| Notes:        | <p>Support for this data set option was added in SAS Viya 3.5.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p>                                                                                                                                                   |
| See:          | <a href="#">DBTYPE= data set option</a>                                                                                                                                                                                                                                     |

---

## Syntax

```

PG_IDENTITY_COLS=(column-name="nextval('sequence-name')"
< ... column-nameN="nextval('sequence-nameN')">

```

## Required Arguments

### ***column-name***

specifies the column name to use for sequential values.

### ***sequence-name***

specifies the name of the sequence to use in the associated column.

## Details

If you create a data set using this option and you do not specify a data type for the associated column, then SAS/ACCESS creates a BIGINT column that cannot take NULL values. You can create a column with a different data type by using the DBTYPE= data set option when you create the data set.

## Example

This example assigns the sequence object Myseq to the Idnum column in table User.Mydata that is being created in the PostgreSQL database. The source for the Mydata table is the SAS data set Work.New.

```
data work.new;
    idnum=.; myname='test1'; output;
    idnum=.; myname='test2'; output;
    idnum=.; myname='test3'; output;
    idnum=.; myname='test4'; output;
    idnum=.; myname='test5'; output;
run;

proc sql;
    create table user.mydata(idnum int, myname char(10));
    insert into user. mydata(bulkload=no
                            pg_identity_cols=(idnum="nextval('myseq') ")
                           );
    select * from work.new;
    select * from user.mydata;
quit;
```

---

## POST\_DML\_STMT\_OPTS= Data Set Option

Specifies text to append to an INSERT, UPDATE, or DELETE statement.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Access                                                              |
| Default:     | None                                                                     |
| Data source: | Oracle                                                                   |
| Note:        | Support for this data set option was added in SAS 9.4M5.                 |
| See:         | <a href="#">POST_DML_STMT_OPTS= LIBNAME option</a>                       |

---

## Syntax

**POST\_DML\_STMT\_OPTS=value**

## Required Argument

### **value**

specifies text to append to an INSERT, UPDATE, or DELETE statement. SAS/ACCESS does not verify the validity of the text that you provide.

## Details

Typically, you use POST\_DML\_STMT\_OPTS= for logging errors that might occur during data manipulation (DML). In this case, you must have created an Oracle log table either outside SAS or by using explicit SQL code with PROC SQL. The log table records errors that might occur during data manipulation (DML). The text that you provide for the POST\_DML\_STMT\_OPTS= LIBNAME option should take this form (information within '< >' is optional):

```
LOG ERRORS <INTO <schema.>table-name> <('simple-expression')>
      <REJECT LIMIT integer | UNLIMITED>
```

This option acts only on the table that immediately follows 'INSERT INTO', 'UPDATE', or 'DELETE FROM' in your PROC SQL query.

To see the SQL code that is generated, use the SASTRACE and SASTRACELOG system options, as shown here:

```
option sastraceloc=saslog sastrace=',,d';
```

Here is an example that uses the POST\_DML\_STMT\_OPTS= data set option to log an unlimited number of insert errors into the table ERR\_ORATAB:

```
proc sql;
  insert into oralib1.ORATAB (post_dml_stmt_opts = 'LOG ERRORS INTO ERR_ORATAB
                                REJECT LIMIT UNLIMITED')
    select *
    from oralib1.CLASS;
quit;
```

The example above generates the following SQL code:

```
insert into ORATAB ("NAME", "SEX", "AGE", "HEIGHT", "WEIGHT")
select TXT_2."NAME", TXT_2."SEX", TXT_2."AGE", TXT_2."HEIGHT", TXT_2."WEIGHT"
from CLASS TXT_2 LOG ERRORS INTO ERR_ORATAB REJECT LIMIT UNLIMITED
```

## POST\_STMT\_OPTS= Data Set Option

Allows additional database-specific options to be placed after the CREATE TABLE statement in generated SQL code.

|           |                                                                            |
|-----------|----------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when creating DBMS tables using SAS/ACCESS software). |
| Category: | Data Set Control                                                           |
| Alias:    | DBCREATE_TABLE_OPTS=                                                       |
| Default:  | none                                                                       |

|              |                                                                                                                                                                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Restriction: | <i>Hadoop</i> : The value that you specify for this option cannot contain a PARTITIONED BY clause.                                                                                                                                                            |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, Netezza, Oracle, PostgreSQL, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica                                        |
| Notes:       | Support for this option was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for JDBC was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS. |
| See:         | <a href="#">DBCREATE_TABLE_OPTS=</a> data set option, <a href="#">DBIDIRECTEXEC</a> system option, <a href="#">POST_TABLE_OPTS=</a> data set option, <a href="#">PRE_STMT_OPTS=</a> data set option, <a href="#">PRE_TABLE_OPTS=</a> data set option          |

## Syntax

**POST\_STMT\_OPTS='DBMS-SQL-options'**

## Required Argument

### DBMS-SQL-option(s)

specifies database-specific options to be placed after the CREATE TABLE statement. Enclose the options that you specify within single or double quotation marks.

## Details

You can use the POST\_STMT\_OPTS= data set option with these related data set options: PRE\_STMT\_OPTS, PRE\_TABLE\_OPTS=, and POST\_TABLE\_OPTS=. For example, you can provide database options according to this template:

```
proc sql;
  create table crttab1 ( POST_TABLE_OPTS= "/* post_table_hint */"
    PRE_TABLE_OPTS= "/* pre_table_hint */"
    POST_STMT_OPTS= "/* post_stmt_hint */"
    PRE_STMT_OPTS= "/* pre_stmt_hint */"
  ) as
  select * from rdtab;
  <additional-clauses>
quit;
```

The resulting code varies depending on whether the DBIDIRECTEXEC system option is enabled. When DBIDIRECTEXEC is specified, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```
/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.crttab1
/* post_table_hint */ as ( select TXT_1."C1", TXT_1."D1",
TXT_1."E1"
  from DBID20.RDTAB TXT_1 ) WITH NO DATA IN UDBID20.USERDATA
/* post_stmt_hint */
```

When you specify NODBIDIRECTEXEC, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```
/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.CRTAB2
  /* post_table_hint */ (C1 FLOAT, D1 CHAR(1), E1 CHAR(1)) IN
  UDBID20.USERDATA
  /* post_stmt_hint */
```

Another usage template that does not use the SQL procedure is shown in this code:

```
data mylib.crtab1 ( POST_TABLE_OPTS= /* post_table_hint */
                     PRE_TABLE_OPTS= /* pre_table_hint */
                     POST_STMT_OPTS= /* post_stmt_hint */
                     PRE_STMT_OPTS= /* pre_stmt_hint */
                   );
set work.localtable;
run;
```

## POST\_TABLE\_OPTS= Data Set Option

Allows additional database-specific options to be placed after the table name in a CREATE TABLE statement.

|              |                                                                                                                                                                                                                                                                                  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating DBMS tables using SAS/ACCESS software).                                                                                                                                                                                                       |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                 |
| Default:     | none                                                                                                                                                                                                                                                                             |
| Interaction: | For Hadoop, if DBCREATE_TABLE_OPTS= or POST_STMT_OPTS= is specified in a LIBNAME statement, then the value for POST_TABLE_OPTS= data set option is ignored.                                                                                                                      |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, Netezza, Oracle, PostgreSQL, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica                                                           |
| Notes:       | <p>Support for this option was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> |
| See:         | <a href="#">DBIDIRECTEXEC system option</a> , <a href="#">POST_STMT_OPTS= data set option</a> , <a href="#">PRE_STMT_OPTS= data set option</a> , <a href="#">PRE_TABLE_OPTS= data set option</a>                                                                                 |

## Syntax

**POST\_TABLE\_OPTS=***DBMS-SQL-options*

## Required Argument

### **DBMS-SQL-option(s)**

specifies additional database-specific options to be placed after the table name in a CREATE TABLE statement.

## Details

You can use the POST\_TABLE\_OPTS= data set option with these related data set options: PRE\_STMT\_OPTS, POST\_STMT\_OPTS=, and PRE\_TABLE\_OPTS=. For example, you can provide database options according to this template:

```
proc sql;
  create table crttab1 ( POST_TABLE_OPTS= /* post_table_hint */
    PRE_TABLE_OPTS= /* pre_table_hint */
    POST_STMT_OPTS= /* post_stmt_hint */
    PRE_STMT_OPTS= /* pre_stmt_hint */
  ) as
    select * from rdtab;
  <additional-clauses>
quit;
```

The resulting code varies depending on whether the DBIDIRECTEXEC system option is enabled. When DBIDIRECTEXEC is specified, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```
/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.crttab1
  /* post_table_hint */ as ( select TXT_1."C1", TXT_1."D1",
  TXT_1."E1"
    from DBID20.RDTAB TXT_1 ) WITH NO DATA IN UDBID20.USERDATA
  /* post_stmt_hint */
```

When you specify NODBIDIRECTEXEC, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```
/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.CRTAB2
  /* post_table_hint */ (C1 FLOAT, D1 CHAR(1), E1 CHAR(1)) IN
  UDBID20.USERDATA
  /* post_stmt_hint */
```

Another usage template that does not use the SQL procedure is shown in this code:

```
data mylib.crttab1 ( POST_TABLE_OPTS= /* post_table_hint */
  PRE_TABLE_OPTS= /* pre_table_hint */
  POST_STMT_OPTS= /* post_stmt_hint */
  PRE_STMT_OPTS= /* pre_stmt_hint */
);
set work.localtable;
run;
```

---

## PRE\_STMT\_OPTS= Data Set Option

Allows additional database-specific options to be placed before a CREATE TABLE statement.

Valid in: DATA and PROC steps (when creating DBMS tables using SAS/ACCESS software).

Category: Data Set Control

Default: none

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, Netezza, Oracle, PostgreSQL, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica

- Notes:
- Support for this option was added in SAS 9.4M3.
  - Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.
  - Support for JDBC was added in SAS Viya 3.4.
  - Support for Snowflake was added in the August 2019 release of SAS/ACCESS.
- See:
- [DBIDIRECTEXEC system option](#), [POST\\_STMT\\_OPTS= data set option](#), [POST\\_TABLE\\_OPTS= data set option](#), [PRE\\_TABLE\\_OPTS= data set option](#)

## Syntax

**PRE\_STMT\_OPTS=***DBMS-SQL-options*

### Required Argument

#### **DBMS-SQL-option(s)**

specifies additional database-specific options to be placed before the CREATE TABLE statement.

## Details

You can use the PRE\_STMT\_OPTS= data set option with these related data set options: PRE\_TABLE\_OPTS, POST\_STMT\_OPTS=, and POST\_TABLE\_OPTS=. For example, you can provide database options according to this template:

```
proc sql;
  create table crttabl ( POST_TABLE_OPTS= /* post_table_hint */
                         PRE_TABLE_OPTS= /* pre_table_hint */
                         POST_STMT_OPTS= /* post_stmt_hint */
                         PRE_STMT_OPTS= /* pre_stmt_hint */
                         ) as
  select * from rdtab;
  <additional-clauses>
quit;
```

The resulting code varies depending on whether the DBIDIRECTEXEC system option is enabled. When DBIDIRECTEXEC is specified, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```
/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.crtabl
/* post_table_hint */ as ( select TXT_1."C1", TXT_1."D1",
TXT_1."E1"
from DBID20.RDTAB TXT_1 ) WITH NO DATA IN UDBID20.USERDATA
/* post_stmt_hint */
```

When you specify NODBIDIRECTEXEC, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```
/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.CRTAB2
/* post_table_hint */ (C1 FLOAT, D1 CHAR(1), E1 CHAR(1)) IN
UDBID20.USERDATA
/* post_stmt_hint */
```

Another usage template that does not use the SQL procedure is shown in this code:

```

data mylib.crtab1 ( POST_TABLE_OPTS= /* post_table_hint */
                     PRE_TABLE_OPTS= /* pre_table_hint */
                     POST_STMT_OPTS= /* post_stmt_hint */
                     PRE_STMT_OPTS= /* pre_stmt_hint */
                  );
set work.localtable;
run;

```

## PRE\_TABLE\_OPTS= Data Set Option

allows additional database-specific options to be placed before the table name in a `CREATE TABLE` statement.

|              |                                                                                                                                                                                                                                                                                  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating DBMS tables using SAS/ACCESS software).                                                                                                                                                                                                       |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                 |
| Default:     | none                                                                                                                                                                                                                                                                             |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, Netezza, Oracle, PostgreSQL, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica                                                           |
| Notes:       | <p>Support for this option was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for JDBC was added in SAS Viya 3.4.</p> <p>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.</p> |
| See:         | <a href="#">DBIDIRECTEXEC system option</a> , <a href="#">POST_STMT_OPTS= data set option</a> ,<br><a href="#">POST_TABLE_OPTS= data set option</a> , <a href="#">PRE_STMT_OPTS= data set option</a>                                                                             |

# Syntax

**PRE\_TABLE\_OPTS=***DBMS-SQL-options*

## Required Argument

## DBMS-SQL-option(s)

specifies additional database-specific options to be placed before the table name in a `CREATE TABLE` statement.

## Details

You can use the PRE\_TABLE\_OPTS= data set option with these related data set options: PRE\_STMT\_OPTS, POST\_STMT\_OPTS=, and POST\_TABLE\_OPTS=. For example, you can provide database options according to this template:

```

POST_STMT_OPTS= /* post_stmt_hint */
PRE_STMT_OPTS= /* pre_stmt_hint */
) as
  select * from rdtab;
<additional-clauses>
quit;

```

The resulting code varies depending on whether the DBIDIRECTEXEC system option is enabled. When DBIDIRECTEXEC is specified, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```

/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.crtab1
  /* post_table_hint */ as ( select TXT_1."C1", TXT_1."D1",
    TXT_1."E1"
      from DBID20.RDTAB TXT_1 ) WITH NO DATA IN UDBID20.USERDATA
  /* post_stmt_hint */

```

When you specify NODBIDIRECTEXEC, the code might result in this generated SQL code (assuming the columns C1, D1, and E1):

```

/* pre_stmt_hint */ CREATE /* pre_table_hint */ TABLE DBID20.CRTAB2
  /* post_table_hint */ (C1 FLOAT, D1 CHAR(1), E1 CHAR(1)) IN
UDBID20.USERDATA
  /* post_stmt_hint */

```

Another usage template that does not use the SQL procedure is shown in this code:

```

data mylib.crtab1 ( POST_TABLE_OPTS= /* post_table_hint */
  PRE_TABLE_OPTS= /* pre_table_hint */
  POST_STMT_OPTS= /* post_stmt_hint */
  PRE_STMT_OPTS= /* pre_stmt_hint */
);
set work.localtable;
run;

```

## PRESERVE\_COL\_NAMES= Data Set Option

Preserves spaces, special characters, and case sensitivity in DBMS column names when you create DBMS tables.

Valid in: DATA and PROC steps (when creating DBMS tables using SAS/ACCESS software).

Category: Data Set Control

Alias: QUOTE NAMES= [Impala, SAP IQ]

Default: LIBNAME option value

Interaction: If you use the DS2 or FedSQL language, quoting and casing of names is different. For more information, see the identifiers topic in [SAS DS2 Language Reference](#) or [SAS FedSQL Language Reference](#).

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Notes: Support for HAWQ was added in SAS 9.4M3.

Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for JDBC was added in SAS Viya 3.4.

Support for Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Spark was added in SAS 9.4M7.

Support for Yellowbrick was added in SAS 9.4M7.

See:

[PRESERVE\\_COL\\_NAMES= LIBNAME option](#), [PRESERVE\\_TAB\\_NAMES= LIBNAME option](#), [VALIDVARNAME= system option](#), [SAS Names and Support for DBMS Names](#), and the DBMS-specific naming conventions sections for your SAS/ACCESS interface

## Syntax

**PRESERVE\_COL\_NAMES=YES | NO**

### Syntax Description

#### YES

specifies that column names that are used in table creation are passed to the DBMS with special characters and the exact, case-sensitive spelling of the name are preserved.

#### NO

specifies that column names that are used in DBMS table creation are derived from SAS variable names by using the SAS variable name normalization rules. (For more information see the VALIDVARNAME= system option.) However, the database applies its DBMS-specific normalization rules to the SAS variable names when it creates the DBMS column names.

The use of name literals to create column names that use database keywords or special symbols other than the underscore character might be invalid when DBMS normalization rules are applied. To include nonstandard SAS symbols or database keywords, specify PRESERVE\_COL\_NAMES=YES.

## Details

This option applies only when you use SAS/ACCESS to create a new DBMS table. When you create a table, you assign the column names by using one of these methods:

- To control the case of the DBMS column names, specify variables with the desired case and set PRESERVE\_COL\_NAMES=YES. If you use special symbols or blanks, you must specify VALIDVARNAME=ANY and use name literals. For more information, see the naming topic in this document and also [SAS Data Set Options: Reference](#).

**SAP HANA:** When you specify PRESERVE\_COL NAMES=YES, you can use reserved words for column names.

- To enable the DBMS to normalize the column names according to its naming conventions, specify variables with any case and set PRESERVE\_COL NAMES=NO.

When you use SAS/ACCESS to read from, insert rows into, or modify data in an existing DBMS table, SAS identifies the database column names by their spelling.

Therefore, when the database column exists, the case of the variable does not matter.

For more information, see the SAS/ACCESS naming topic in the DBMS-specific reference section for your interface.

To use column names in your SAS program that are not valid SAS names, you must use one of these techniques.

- Use the DQUOTE= option in PROC SQL and reference your columns using double quotation marks. Here is an example.

```
proc sql dquote=ansi;
    select "Total$Cost" from mydblib.mytable;
```

- Specify the global VALIDVARNAME=ANY system option and use name literals in the SAS language. Here is an example.

```
proc print data=mydblib.mytable;
    format 'Total$Cost'n 22.2;
```

If you are *creating* a table in PROC SQL, you must also include the PRESERVE\_COL\_NAMES=YES option. Here is an example.

```
libname mydblib hadoop user=myusr1 password=mypwd1 server=hadoopsvr;
proc sql dquote=ansi;
    create table mydblib.mytable (preserve_col_names=yes) ("my$column" int);
```

PRESERVE\_COL\_NAMES= does not apply to the SQL pass-through facility.

## QUALIFIER= Data Set Option

Specifies the qualifier to use when you are reading database objects, such as DBMS tables and views.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: LIBNAME option value

Data source: Amazon Redshift, Google BigQuery, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, PostgreSQL, Vertica

Notes: Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for Google BigQuery was added in the August 2019 release of SAS/ACCESS.

See: [QUALIFIER= LIBNAME option](#)

## Syntax

**QUALIFIER=<qualifier-name>**

---

## Details

If this option is omitted, the default qualifier name, if any, is used for the data source. **QUALIFIER=** can be used for any data source, such as a DBMS object, that allows three-part identifier names: *qualifier.schema.object*.

---

## QUERY\_BAND= Data Set Option

Specifies whether to set a query band for the current transaction.

|              |                                                                                                                                                                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                        |
| Default:     | none                                                                                                                                                                                                                                                                                    |
| Data source: | Teradata                                                                                                                                                                                                                                                                                |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a><br><a href="#">BULKLOAD= data set option</a><br><a href="#">FASTEXPORT= LIBNAME option</a><br><a href="#">Maximizing Teradata Load Performance</a><br><a href="#">MULTILOAD= data set option</a><br><a href="#">QUERY_BAND= LIBNAME option</a> |

---

## Syntax

**QUERY\_BAND="pair-name=pair\_value;"**

### Syntax Description

**"pair-name=pair\_value;"**

specifies a name and value pair of a query band for the current transaction.

---

## Details

Use this option to set unique identifiers on Teradata transactions and to add them to the current transaction. The Teradata engine uses this syntax to pass the name-value pair to Teradata:

```
data db.new_table (QUERY_BAND="org=Sales; process=Build_Initial;");
      set work.sasdata1;
run;
```

For more information about this option and query-band limitations, see *Teradata SQL Reference: Data Definition Statements*.

---

## QUERY\_TIMEOUT= Data Set Option

Specifies the number of seconds of inactivity to wait before canceling a query.

|              |                                                                                                                                                                                                                                                                                                                                                                   |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                          |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                  |
| Alias:       | TIMEOUT= [Greenplum, HAWQ, Microsoft SQL Server, SAP IQ]                                                                                                                                                                                                                                                                                                          |
| Default:     | LIBNAME option value                                                                                                                                                                                                                                                                                                                                              |
| Interaction: | Set the SAS SQLIPONEATTEMPT system option to disable PROC SQL from trying the query again after the first query times out.                                                                                                                                                                                                                                        |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, Netezza, ODBC, PostgreSQL, SAP HANA, SAP IQ, Snowflake, Spark, Vertica, Yellowbrick                                                                                                                                                             |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Hadoop and JDBC was added in SAS Viya 3.4.<br>Support for Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Spark was added in SAS 9.4M7.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">QUERY_TIMEOUT= LIBNAME option</a> , <a href="#">SQLIPONEATTEMPT system option</a>                                                                                                                                                                                                                                                                     |

---

## Syntax

**QUERY\_TIMEOUT=***number-of-seconds*

---

## Details

QUERY\_TIMEOUT= 0 indicates that there is no time limit for a query. This option is useful when you are testing a query, you suspect that a query might contain an endless loop, or the data is locked by another user.

---

## READBUFF= Data Set Option

Specifies the number of rows of DBMS data to read into the buffer.

|           |                                                                          |
|-----------|--------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category: | Data Set Control                                                         |
| Aliases:  | BUFF=, BUFFSIZE= [Oracle]                                                |

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | ROWSET_SIZE= [Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Impala, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|               | ROWSET= [SAP IQ]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Default:      | LIBNAME option value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Restrictions: | <p>When READBUFF=1, only one row is retrieved at a time.</p> <p>Buffering data reads can decrease network activities and increase performance. However, because SAS stores the rows in memory, higher values for READBUFF= use more memory. In addition, if too many rows are selected at once, rows that are returned to the SAS application might be out of date. For example, if someone else modifies the rows, you do not see the changes.</p> <p>[Oracle] The default is 250 rows per fetch. Although this value can be up to 2,147,483,647 rows per fetch, this depends on available memory. A practical limit for most applications is less, so the total maximum buffer size must not exceed 2GB.</p> |
| Interaction:  | Hadoop: This option is applied only when READ_METHOD=JDBC.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Data source:  | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, JDBC, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Vertica, Yellowbrick                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Notes:        | <p>Support for HAWQ was added in SAS 9.4M3.</p> <p>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.</p> <p>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.</p> <p>Support for Yellowbrick was added in SAS 9.4M7.</p> <p>Support for Hadoop, JDBC, and Spark was added in SAS 9.4M8.</p>                                                                                                                                                                                                                                                                                                                                           |
| Tips:         | <p>This option improves performance by specifying a number of rows that can be held in memory for input into SAS.</p> <p>The higher the value for READBUFF=, the more rows that the engine retrieves in one fetch operation.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| See:          | <a href="#">INSERTBUFF= LIBNAME option</a> , <a href="#">INSERTBUFF= data set option</a> , <a href="#">READBUFF= LIBNAME option</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

---

## Syntax

**READBUFF=***integer*

### Syntax Description

#### *integer*

the positive number of rows to hold in memory. SAS allows the maximum number that the DBMS allows.

---

## READ\_ISOLATION\_LEVEL= Data Set Option

Specifies which level of read isolation locking to use when you are reading data.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                         |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Alias:       | RIL= [Greenplum, HAWQ, Microsoft SQL Server, SAP IQ]                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Default:     | LIBNAME option value                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Interaction: | This option is ignored if you do not specify READ_LOCK_TYPE=ROW. [DB2 under UNIX and PC Hosts, Netezza, ODBC]                                                                                                                                                                                                                                                                                                                                                                    |
| Data source: | DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, SAP ASE, Teradata, Vertica                                                                                                                                                                                                                                                                                                                                             |
| See:         | <a href="#">READ_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= data set option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= data set option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a> , <a href="#">UPDATE_LOCK_TYPE= data set option</a> , and DBMS-specific locking information in the reference section for your SAS/ACCESS interface |

---

## Syntax

**READ\_ISOLATION\_LEVEL=DBMS-specific-value**

### Syntax Description

***dbms-specific-value***

See the DBMS-specific reference section for your interface for this value.

---

## Details

The degree of isolation specifies the degree to which these items are affected:

- Rows that the current application reads and updates are available to other concurrently executing applications.
- Update activity of other concurrently executing application processes can affect the current application.

---

## READ\_LOCK\_TYPE= Data Set Option

Specifies how data in a DBMS table is locked during a read transaction.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                               |
| Category:    | Data Access                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Defaults:    | LIBNAME option value [DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica, Yellowbrick]<br>ROW [Amazon Redshift]                                                                                                                                                                                                                                                        |
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                                                                                                       |
| Notes:       | Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                                                                                                                                                                                                      |
| Tip:         | If you omit READ_LOCK_TYPE=, you receive either the default action for the DBMS that you are using or a lock for the DBMS that was specified with the LIBNAME statement.                                                                                                                                                                                                                                                                                                               |
| See:         | <a href="#">READ_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">READ_ISOLATION_LEVEL= data set option</a> , <a href="#">READ_LOCK_TYPE= LIBNAME option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= data set option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a> , <a href="#">UPDATE_LOCK_TYPE= data set option</a> , and DBMS-specific locking information in the reference section for your SAS/ACCESS interface |

## Syntax

**READ\_LOCK\_TYPE=ROW | PAGE | TABLE | NOLOCK | VIEW**

### Syntax Description

#### ROW

locks a row if any of its columns are accessed. If you are using the interface to ODBC, Teradata, or DB2 under UNIX and PC Hosts, READ\_LOCK\_TYPE=ROW indicates that locking is based on the READ\_ISOLATION\_LEVEL= LIBNAME option.

**Data source** DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Microsoft SQL Server, ODBC, Oracle, PostgreSQL, SAP HANA, SAP IQ, Teradata, Vertica

#### PAGE

locks a page of data, which is a DBMS-specific number of bytes.

**Data source** SAP ASE

#### TABLE

locks the entire DBMS table. If you specify READ\_LOCK\_TYPE=TABLE, you must also specify the CONNECTION=UNIQUE, or you receive an error message. Specifying CONNECTION=UNIQUE ensures that your table lock is not lost (for example, due to another table closing and committing rows in the same connection).

Data source DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, Oracle, SAP IQ, Teradata

**NOLOCK**

does not lock the DBMS table, pages, or any rows during a read transaction.

Data source Microsoft SQL Server, ODBC with Microsoft SQL Server driver, OLE DB, Oracle, SAP ASE

**VIEW**

locks the entire DBMS view.

Data source Teradata

---

## READ\_METHOD= Data Set Option

Specifies how to read data.

|              |                                                                                                                                  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                       |
| Category:    | Data Set Control                                                                                                                 |
| Default:     | none                                                                                                                             |
| Data source: | Hadoop, Spark                                                                                                                    |
| Note:        | Support for Spark was added in SAS 9.4M7.                                                                                        |
| See:         | <a href="#">ANALYZE= LIBNAME option</a> , <a href="#">ANALYZE= data set option</a> , <a href="#">READ_METHOD= LIBNAME option</a> |

---

## Syntax

**READ\_METHOD=**[JDBC](#) | [HDFS](#)

### Syntax Description

**JDBC**

specifies that data is to be read through the JDBC connection to the Hive service. You can use the ANALYZE= option to potentially improve performance when querying small tables.

**HDFS**

specifies that data is to be read through a connection to the Hadoop HDFS service.

## Details

Although HDFS cannot alter the behavior of operations that always use JDBC, in general HDFS is a faster alternative to JDBC. To take advantage of potential performance benefits, set this option to HDFS. Use JDBC when you cannot access the HDFS service or JDBC Read offers some other advantage.

## Example: Read Data Using JDBC

In this example, a partition of data from the sales Hive table is read using JDBC.

```
libname hdp hadoop server=mysrv1 user=myusrl pwd=mypwd1;
data work.sales_subset; set hdp.sales(READ_METHOD=JDBC);
where year_month='2012-10'; run;
```

---

## READ\_MODE= Data Set Option

Specifies the method to use to move data into SAS.

|              |                                                                                                                   |
|--------------|-------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                          |
| Category:    | Data Set Control                                                                                                  |
| Default:     | LIBNAME option value                                                                                              |
| Data source: | Google BigQuery                                                                                                   |
| Note:        | Support for this data set option was added in the April 2021 update for SAS/ACCESS on SAS 9.4M7 and SAS Viya 3.5. |
| Tip:         | The STORAGE value is recommended if the Google Storage API is available for your Google BigQuery data source.     |
| See:         | <a href="#">READ_MODE= LIBNAME option</a>                                                                         |

---

## Syntax

**READ\_MODE=STANDARD | STORAGE**

### Required Argument

#### **STANDARD | STORAGE**

specifies the method to use when moving data from Google BigQuery into SAS.

STANDARD    use the default SAS/ACCESS method to move data into SAS.

STORAGE    use the Storage API for Google to move data into SAS.

---

## READ\_MODE\_WAIT= Data Set Option

Specifies during SAS/ACCESS Read operations whether Teradata waits to acquire a lock or fails your request when a different user has locked the DBMS resource.

|              |                                                                                    |
|--------------|------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)           |
| Category:    | Data Access                                                                        |
| Default:     | LIBNAME option value                                                               |
| Data source: | Teradata                                                                           |
| See:         | <a href="#">READ_MODE_WAIT= LIBNAME option , Locking in the Teradata Interface</a> |

---

## Syntax

**READ\_MODE\_WAIT=YES | NO**

### Syntax Description

#### **YES**

specifies that Teradata waits to acquire the lock, and SAS/ACCESS waits indefinitely until it can acquire the lock.

#### **NO**

specifies that Teradata fails the lock request if the specified DBMS resource is locked.

---

## Details

If you specify READ\_MODE\_WAIT=NO, and a different user holds a *restrictive* lock, then the executing SAS step fails. SAS/ACCESS continues to process the job by executing the next step. If you specify READ\_MODE\_WAIT=YES, SAS/ACCESS waits indefinitely until it can acquire the lock.

A restrictive lock means that another user is holding a lock that prevents you from obtaining your desired lock. Until the other user releases the restrictive lock, you cannot obtain your lock. For example, another user's table-level WRITE lock prevents you from obtaining a READ lock on the table.

For more information, see locking topic in the Teradata section.

---

## ROW\_DELIMITER= Data Set Option

Specifies the single delimiter character to use to separate Hive table records.

|              |                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                               |
| Category:    | Data Set Control                                                                                       |
| Aliases:     | ROW_DELIM<br>ROW_DELIMIT                                                                               |
| Default:     | \010 ('\n')                                                                                            |
| Restriction: | Hive rejects this delimiter if you specify it in any other way, such as ROW_DELIMITER=010 (=linefeed). |
| Requirement: | Specify a single-character or three-digit decimal ASCII value between 001 and 127.                     |
| Data source: | Hadoop                                                                                                 |
| See:         | <a href="#">COLUMN_DELIMITER= data set option</a>                                                      |
| Examples:    | ROW_DELIMITER="#"<br>ROW_DELIMITER=009 (tab)                                                           |

---

## Syntax

**ROW\_DELIMITER=***single-character*

---

## SASDATEFMT= Data Set Option

Changes the SAS date format of a DBMS column.

|              |                                                                                                                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                       |
| Category:    | Data Set Control                                                                                                                                                                                                                                                               |
| Default:     | DBMS-specific                                                                                                                                                                                                                                                                  |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Greenplum, HAWQ, Impala, Informix, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                          |
| Notes:       | Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7. |
| See:         | <a href="#">DBSASTYPE= data set option</a>                                                                                                                                                                                                                                     |

---

## Syntax

**SASDATEFMT=(DBMS-date-col-1='SAS-date-format'  
<... DBMS-date-col-n='SAS-date-format'>)**

## Syntax Description

### **DBMS-date-col**

specifies the name of a date column in a DBMS table.

### **SAS-date-format**

specifies a SAS date format that has an equivalent (like-named) informat. For example, DATETIME21.2 is both a SAS format and a SAS informat, so it is a valid value for the **SAS-date-format** argument.

## Details

If the SAS column date format does not match the date format of the corresponding DBMS column, convert the SAS date values to the appropriate DBMS date values. Use the **SASDATEFMT=** option to convert date values from the default SAS date format to another SAS date format that you specify.

Use the **SASDATEFMT=** option to prevent date type mismatches in these circumstances:

- during input operations to convert DBMS date values to the correct SAS DATE, TIME, or DATETIME values
- during output operations to convert SAS DATE, TIME, or DATETIME values to the correct DBMS date values.

The column names specified in this option must be DATE, DATETIME, or TIME columns; columns of any other type are ignored.

The format specified must be a valid date format; output with any other format is unpredictable.

If the SAS date format and the DBMS date format match, this option is not needed.

The default SAS date format is DBMS-specific and is determined by the data type of the DBMS column. See the DBMS-specific reference section about data types for your SAS/ACCESS interface.

**Note:** For non-English date types, SAS automatically converts the data to the SAS type NUMBER. **SASDATEFMT=** does not currently handle these date types.

However, you can use a PROC SQL view to convert the DBMS data to a SAS date format as you retrieve the data or use a format statement in other contexts.

**Oracle:** It is recommended that you use **DBSASTYPE=** instead of **SASDATEFMT=**.

## Examples

### Example 1: Change the Date Format in Oracle

In this example, the APPEND procedure adds SAS data from the SASLIB.DELAY data set to the Oracle table that is accessed by MYDBLIB.INTERNAT. Using **SASDATEFMT=**, the default SAS format for the Oracle column DATES is changed to the DATE9. format. Data output from SASLIB.DELAY into the DATES column in

MYDBLIB.INTERNAT now converts from the DATE9. format to the Oracle format assigned to that type.

```
libname mydblib oracle user=myusr1 password=mypwd1;
libname saslib 'your-SAS-library';
proc append base=mydblib.internat(sasdatefmt=(dates='date9.')) force
  data=saslib.delay;
run;
```

## Example 2: Change a SAS Date Format to a Teradata Format

In the next example, SASDATEFMT= converts DATE1, a SAS DATETIME value, to a Teradata date column named DATE1.

```
libname x teradata user=myusr1 password=mypwd1;
proc sql noerrorstop;
  create table x.dateinfo ( date1 date );
  insert into x.dateinfo
  ( sasdatefmt=( date1='datetime21.' ) )
  values ( '31dec2000:01:02:30'dt );
```

## Example 3: Change a Teradata Date Format to a SAS Format

In this example, SASDATEFMT= converts DATE1, a Teradata date column, to a SAS DATETIME type named DATE1.

```
libname x teradata user=myusr1 password=mypwd1;
data sas_local;
format date1 datetime21.;
set x.dateinfo( sasdatefmt=( date1='datetime21.' ) );
run;
```

## SCANSTRINGCOLUMNS= Data Set Option

Specifies whether to determine the maximum length of VARCHAR columns in a database table or query.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Alias: SCAN\_STRING\_COLUMNS=,SCAN\_STRINGS=, SCANSTRINGS=

Default: LIBNAME option value

Data source: Google BigQuery

Note: Support for this option was added in the April 2021 updates for SAS/ACCESS on SAS 9.4M7 and SAS Viya 3.5.

See: [SCANSTRINGCOLUMNS= LIBNAME option](#)

## Syntax

**SCANSTRINGCOLUMNS=YES | NO**

### Required Arguments

#### **YES**

specifies to scan all VARCHAR columns in a database table to determine the actual maximum length of the columns before loading data. If the VARCHAR columns have been created with the maximum VARCHAR precision, then using this option can reduce the size of the resulting table and accelerate the loading process.

#### **NO**

specifies that VARCHAR columns are not scanned before loading data.

---

## Details

This option can be specified for any table, but the best performance improvement occurs for larger tables. A table is considered to be large if it contains a large number of rows, a large number of VARCHAR columns, or both.

---

## SCHEMA= Data Set Option

Specifies the schema to use when accessing tables and views in a database.

|              |                                                                                                                                                                                                                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                  |
| Category:    | Data Access                                                                                                                                                                                                                                                                                                               |
| Aliases:     | DATABASE= [Impala]<br>OWNER= [Greenplum, HAWQ, Microsoft SQL Server, SAP IQ, Snowflake]                                                                                                                                                                                                                                   |
| Defaults:    | LIBNAME option value [Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, SAP ASE, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick]<br>AUTHID= [DB2 under z/OS]                          |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Teradata, Vertica, Yellowbrick                                              |
| Notes:       | Support for Netezza was added in SAS 9.4M2.<br>Support for HAWQ was added in SAS 9.4M3.<br>Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for JDBC was added in SAS Viya 3.4.<br>Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS. |

Support for Yellowbrick was added in SAS 9.4M7.

See: [DBCONINIT= LIBNAME option](#), [PRESERVE\\_TAB\\_NAMES= LIBNAME option](#), [SCHEMA= LIBNAME option](#)

## Syntax

**SCHEMA=schema-name**

## Syntax Description

### **schema-name**

specifies the name that is assigned to a logical classification of objects in a relational database.

If the schema name contains spaces or non-alphanumeric characters, enclose the value in quotation marks.

## Details

For this option to work, you must have appropriate privileges to access the schema that is specified.

If you do not specify this option, you connect to any schema in the default database for your DBMS. When you specify SCHEMA=, this option acts as a filter to access only the tables and views that belong to that schema.

*Aster:* The default is `none`, which uses the database user's default schema. When the user's default schema is the user name, the user name is used instead. An example is when SQLTables is called to obtain a table listing using PROC DATASETS or SAS Explorer.

*DB2 under z/OS:* If you specify `DBCONINIT="SET CURRENT SQLID='user-ID'"`, then any value that is specified for SCHEMA= is ignored.

*Informix:* The SCHEMA= data set option disables implicit pass-through.

*Netezza:* Starting in Netezza 7.0.3, you can configure your Netezza server for multiple schema support. The Netezza System Administrator's Guide describes how to provision schema support. Multiple schema support is not enabled by default. The default is the database user's default schema.

*Oracle:* The default is the LIBNAME value. If PRESERVE\_TAB\_NAMES=NO, SAS converts the SCHEMA= value to uppercase because all values in the Oracle data dictionary are converted to uppercase unless quoted.

*SAP ASE:* You cannot use the SCHEMA= option when you use UPDATE\_LOCK\_TYPE=PAGE to update a table.

*Teradata:* The default is the LIBNAME value, if specified. You can use this option to point to a database that is different from your default database. Using the SCHEMA= option does not establish a physical connection to the specified schema. This option lets you view or modify a different user's DBMS tables or views if you have the required Teradata privileges. For example, to read another user's tables, you must have the Teradata privilege SELECT for that user's tables.

## Example

In this example, SCHEMA= causes DB2 to interpret Mydb.Temp\_Emps as Scott.Temp\_Emps.

```
proc print data=mydb.temp_emps
  schema=SCOTT;
run;
```

In this next example, SAS sends any reference to Employees as Scott.Employees.

```
libname mydblib oracle user=myusr1 password=mypwd1 path="myorapath";
proc print data=employees (schema=scott);
run;
```

In this example, user MYUSR1 prints the contents of the Employees table, which is located in the Donna database.

```
libname mydblib teradata user=myusr1 pw=mypwd1;
proc print data=mydblib.employees(schema=donna);
run;
```

---

## SCRATCH\_DB= Data Set Option

Specifies a Hive schema so that SAS can store temporary output.

|              |                                                 |
|--------------|-------------------------------------------------|
| Valid in:    | SAS/ACCESS LIBNAME statement, CONNECT statement |
| Category:    | Data Set Control                                |
| Default:     | none (the current schema is used)               |
| Requirement: | SAS must be able to write to the schema.        |
| Data source: | Hadoop, Impala                                  |
| Note:        | Support for this option was added in SAS 9.4M4. |
| See:         | <a href="#">SCRATCH_DB= LIBNAME option</a>      |

---

## Syntax

**SCRATCH\_DB=***schema-name*

### Syntax Description

#### ***schema-name***

specifies the name that is assigned to a logical classification of objects in a relational database.

## Details

This option lets the user specify a target schema that SAS can use for temporary output. It is needed when a user does not have permissions to perform DDL operations such as CREATE TABLE or DROP TABLE in the current schema.

---

## SEGMENT\_NAME= Data Set Option Data Set Option

Lets you control the segment in which you create a table.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Default:     | none                                                                     |
| Data source: | SAP ASE                                                                  |

---

## Syntax

**SEGMENT\_NAME=***segment-name*

### Syntax Description

***segment-name***

specifies the name of the segment in which to create a table.

---

## SESSIONS= Data Set Option

Specifies how many Teradata sessions to be logged on when using FastLoad, FastExport, or Multiload.

|              |                                                                                            |
|--------------|--------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software) |
| Category:    | Bulk Loading                                                                               |
| Default:     | none                                                                                       |
| Data source: | Teradata                                                                                   |
| See:         | <a href="#">SESSIONS= LIBNAME option</a>                                                   |

---

## Syntax

**SESSIONS=***number-of-sessions*

## Syntax Description

### **number-of-sessions**

specifies a numeric value that indicates the number of sessions to be logged on.

## Details

When reading data with FastExport or loading data with FastLoad and MultiLoad, you can request multiple sessions to increase throughput. Using large values might not necessarily increase throughput due to the overhead associated with session management. Check whether your site has any recommended value for the number of sessions to use. See your Teradata documentation for details about using multiple sessions.

## Example: Request Sessions to Load a Table

This example uses SESSIONS= in a LIBNAME statement to request that five sessions be used to load data with FastLoad.

```
libname x teradata user=myusr1 pw=mypwd1;
proc datasets library=x;
  delete test;run;
data x.test(FASTLOAD=YES SESSIONS=5);
  i=5;
run;
```

## SET= Data Set Option

Specifies whether duplicate rows are allowed when creating a table.

|              |                                                                                            |
|--------------|--------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software) |
| Category:    | Data Set Control                                                                           |
| Alias:       | TBLSET                                                                                     |
| Default:     | NO                                                                                         |
| Data source: | Teradata                                                                                   |

## Syntax

**SET=YES | NO**

## Syntax Description

### YES

specifies that no duplicate rows are allowed.

### NO

specifies that duplicate rows are allowed.

## Details

Use the SET= data set option to specify whether to allow duplicate rows when creating a table. This option overrides the default Teradata MULTISET characteristic.

## Example: Create a Table without Duplicate Rows

This example creates a Teradata table of type SET that does not allow duplicate rows.

```
libname trlib teradata user=myusr1 pw=mypwd1;
options sastrace=',,d' sastraceloc=saslog;
proc datasets library=x;
  delete test1;run;
data x.test1(TBLSET=YES);
  i=1;output;
  run;
```

## SLEEP= Data Set Option

Specifies the number of minutes that FastExport, FastLoad, or MultiLoad waits before trying again to log on to Teradata.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                  |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Default:     | 6                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Tip:         | The data set option has precedence over the LIBNAME option.                                                                                                                                                                                                                                                                                                                                                                                                 |
| See:         | <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">FASTEXPORT= LIBNAME option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">SLEEP= LIBNAME option</a> , <a href="#">TENACITY= LIBNAME option</a> , <a href="#">TENACITY= data set option</a> , <a href="#">Using the TPT API</a> |

## Syntax

**SLEEP=***number-of-minutes*

### Syntax Description

#### *number-of-minutes*

the number of minutes to wait before trying again to log on to Teradata.

## Details

Use this option to indicate to FastExport, FastLoad, or MultiLoad how long to wait before it retries logging on to Teradata when the maximum number of utilities are already running. (The maximum number of Teradata utilities that can run concurrently varies from 5 to 15, depending on the database server value.) The default value for SLEEP= is 6 minutes. The value that you specify for SLEEP= must be greater than 0.

Use SLEEP= with TENACITY=, which specifies the time in hours that FastExport, FastLoad, or MultiLoad must continue to try the logon operation. SLEEP= and TENACITY= function very much like the SLEEP and TENACITY run-time options of the native Teradata FastExport, FastLoad, or MultiLoad utility.

---

## SUB\_CHAR= Data Set Option

Specifies a substitution character to use in place of any invalid characters that cannot be represented in the SAS session encoding.

|              |                                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                   |
| Category:    | Data Set Control                                                                                                           |
| Alias:       | SUBCHAR=                                                                                                                   |
| Default:     | LIBNAME option value                                                                                                       |
| Data source: | Amazon Redshift, Greenplum, Hadoop, HAWQ, Impala, JDBC, Netezza, PostgreSQL, SAP HANA, Spark, Vertica                      |
| Notes:       | Support for this data set option was added in SAS Viya 3.5.<br>Support for Hadoop, JDBC, and Spark was added in SAS 9.4M7. |
| See:         | <a href="#">SUB_CHAR= LIBNAME option</a>                                                                                   |

---

## Syntax

**SUB\_CHAR=***value*

## Required Argument

### **value**

specifies the substitution character to use in place of a character that cannot be represented in the SAS session encoding.

Here are the possible values for this option:

|              |                                                                                                                                                                                                                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| QUESTIONMARK | specifies that a question mark ('?') replaces an invalid character during transcoding.                                                                                                                                                                                                                                                                             |
| SPACE        | specifies that a space replaces an invalid character during transcoding.                                                                                                                                                                                                                                                                                           |
| SUB          | specifies that the default substitution character for the source encoding replaces an invalid character during transcoding. The default substitution character varies depending on your source encoding and on your system platform. For example, the substitution character might be represented by a '0x1A' character on Windows and by a '?' on Linux machines. |
| UESC         | specifies that the '\uddd' value replaces an invalid character during transcoding. This UESC value is typically used only during debugging and is not intended to be used for large-scale jobs.                                                                                                                                                                    |

---

## Details

This option applies only when SAS/ACCESS encodes the data into the SAS session encoding when reading data into SAS. If your data is encoded by your DBMS client to the SAS session encoding before loading data into SAS, then this option is not used.

The default SAS session encoding is UTF-8. If your data must be encoded from another encoding to the SAS session encoding and if your data contains characters that cannot be encoded to the destination encoding, then warnings are generated in the SAS log.

To see details about the rows and columns that are affected by encoding errors, enable the **SASTRACE=** option:

```
options sastrace=',,,d';
```

---

## SYSTEM\_TIMEFRAME= Data Set Option

lets you provide a datetime range to use when querying or modifying a temporal table.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Set Control

Default: none

Data source: DB2 for z/OS

See: [TEMPORAL= data set option](#), [OVERLAPS= data set option](#), [BUSINESS\\_DATATYPE= data set option](#), [BUSINESS\\_TIMEFRAME= data set option](#)

## Syntax

**SYSTEM\_TIMEFRAME=FROM *datetime1* TO *datetime2***

### Syntax Description

*datetimeN*

specifies a datetime value that indicates the beginning or end of a time period that is used when you query or modify a table that contains temporal data.

Provide datetime values that include year, month, day, hours, minutes, seconds, and fractions of a second. For example, use the following code to specify the time period from midnight, November 1, 2012 to midnight, December 1, 2030:

```
system_timeframe="from '2012-11-01-00.00.00.0' to '2030-12-01-00.00.00.0'"
```

## TABLE\_TYPE= Data Set Option

Specifies the type of temporary tables and table storage when the engine creates tables in SAP HANA.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Data Access

Default: none

Restrictions: If you do not specify a value for this option, tables that are created in SAP HANA follow the SAP HANA default for row or column store.

ROW and COLUMN are mutually exclusive.

LOCAL | LOCAL TEMPORARY and GLOBAL | GLOBAL TEMPORARY are mutually exclusive.

Do not specify LOCAL | LOCAL TEMPORARY or GLOBAL | GLOBAL TEMPORARY for permanent tables.

Interaction: The TABLE\_TYPE data set option overrides the TABLE\_TYPE LIBNAME option.

Data source: SAP HANA

See: [CONNECTION=](#) (for sharing sessions across librefs), [TABLE\\_TYPE= LIBNAME option](#)

Examples: TABLE\_TYPE=ROW

TABLE\_TYPE=COLUMN

TABLE\_TYPE=LOCAL

TABLE\_TYPE=GLOBAL

TABLE\_TYPE=(COLUMN GLOBAL)

## Syntax

**TABLE\_TYPE=ROW | COLUMN | LOCAL  
| LOCAL TEMPORARY | GLOBAL | GLOBAL TEMPORARY**

### Syntax Description

#### ROW

creates a table using ROW-based storage in SAP HANA.

#### COLUMN

creates a table using COLUMN-based storage in SAP HANA.

#### LOCAL | LOCAL TEMPORARY

creates a local temporary table in SAP HANA. The table definition and data are visible only in the current session.

#### GLOBAL | GLOBAL TEMPORARY

creates a global temporary table in SAP HANA. The global temporary tables are globally available, and the data is visible only in the current session.

---

## Details

This option takes effect when a SAS program creates a table in SAP HANA.

---

## TEMPORAL= Data Set Option

specifies the type of temporal data that is contained in a table.

|              |                                                                                                                                                                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                        |
| Category:    | Data Set Control                                                                                                                                                                                                                                                |
| Default:     | none                                                                                                                                                                                                                                                            |
| Requirement: | DB2 z/OS with SAS 9.4                                                                                                                                                                                                                                           |
| Data source: | DB2 for z/OS                                                                                                                                                                                                                                                    |
| See:         | <a href="#">OVERLAPS= data set option</a> , <a href="#">BUSINESS_DATATYPE= data set option</a> ,<br><a href="#">BUSINESS_TIMEFRAME= data set option</a> , <a href="#">SYSTEM_TIMEFRAME= data set option</a> ,<br>“Temporal Data for DB2 under z/OS” on page 836 |

---

## Syntax

**TEMPORAL=BUSINESS | SYSTEM | BITEMPORAL**

## Syntax Description

### **BUSINESS**

specifies a temporal table that contains business time. The BUSINESS options add the following columns to the end for storing temporal data: BUS\_START and BUS\_END.

### **SYSTEM**

specifies a temporal table that contains system time. The SYSTEM option adds the following columns to the table for storing temporal data: SYS\_START, SYS\_END, and TRANS\_START.

### **BITEMPORAL**

specifies a temporal table that contains business and system time data. The BITEMPORAL option adds the following columns to the table for storing temporal data: SYS\_START, SYS\_END, BUS\_START, BUS\_END, and TRANS\_START.

## TEMPORAL\_QUALIFIER= Data Set Option

Specifies time-dimension criteria for retrieving data from Teradata.

|              |                                                                                            |
|--------------|--------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                   |
| Category:    | Data Set Control                                                                           |
| Defaults:    | CURRENT VALIDTIME (valid-time column)<br>CURRENT TRANSACTIONTIME (transaction-time column) |
| Interaction: | Specifying values in a DATA step overrides LIBNAME values.                                 |
| Data source: | Teradata                                                                                   |
| See:         | <a href="#">TEMPORAL_QUALIFIER= LIBNAME option</a>                                         |

## Syntax

```
TEMPORAL_QUALIFIER=CURRENT VALIDTIME
| VALIDTIME AS OF 'period'
| SEQUENCED VALIDTIME 'period' | NONSEQUENCED VALIDTIME 'period'
TEMPORAL_QUALIFIER=CURRENT TRANSACTIONTIME
| TRANSACTION AS OF 'period'
| NONSEQUENCED TRANSACTIONTIME
```

## Syntax Description

### **CURRENT VALIDTIME**

selects rows that are valid at the current time.

### **VALIDTIME AS OF PERIOD '*period*'**

selects rows with valid time periods that overlap the specified AS OF period. For the period, you can specify either a single date or a time period (date range) by specifying a start date and an end date.

**SEQUENCED VALIDTIME PERIOD '*period*'**

selects history, current, or future rows that are valid for the specified time period.

**NONSEQUENCED VALIDTIME PERIOD '*period*'**

treats the table as nontemporal.

**CURRENT TRANSACTIONTIME**

selects rows that are open in transaction time.

**TRANSACTIONTIME AS OF '*period*'**

selects rows with transaction-time periods that overlap the specified AS OF period. For the period, you can specify either a single date or a time period (date range) by specifying a start date and an end date.

**NONSEQUENCED TRANSACTIONTIME**

treats the table as nontemporal.

## Details

Use temporal qualifiers to specify time criteria for selecting data from temporal tables.

To use them, before the SQL add a value that you specify for one or more temporal qualifiers for a data set. For example, if you specify TEMPORAL\_QUALIFIER='AS OF PERIOD '(1999-01-01, 2099-01-05)' ' in a DATA step, 'AS OF PERIOD '(1999-01-01, 2099-01-05)' ' is added before the SQL to select the data.

## Examples

### Example 1: Select ValidTime Rows at the Current Time

```
/* Consider data as of 1995-01-01. */
libname x teradata user=myusr1 pw=mypwd1 server=mysrv1
      TEMPORAL_QUALIFIER='VALIDTIME AS OF DATE '1995-01-01' '

/* ASOF PERIOD '(1999-01-01, 2099-01-05)' select * from mytest is
submitted. */
proc print data=x.mytest (TEMPORAL_QUALIFIER='CURRENT VALIDTIME');
run;
```

### Example 2: Select ValidTime Rows for a Specific Date

This example extracts salary details for employees who worked on January 1, 2000.

Employment data from this employee table contains the ValidTime data column, JobDuration.

---

| EName | E_Id | Dept  | Job_duration              |
|-------|------|-------|---------------------------|
| Sania | 1001 | Dept1 | 1990-01-01, 2003-01-01    |
| Sania | 1001 | Dept3 | 2003-01-01, UNTIL_CHANGED |
| Ash   | 1002 | Dept1 | 1995-01-01, 2000-01-01    |
| Ash   | 1002 | Dept2 | 1999-01-01, 2010-01-01    |

Salary data is from the ValidTime column, SalaryPeriod.

| E_Id | Sal   | SalaryPeriod              |
|------|-------|---------------------------|
| 1001 | 10000 | 1990-01-01, 2003-01-01    |
| 1001 | 20000 | 2003-01-01, 2010-01-01    |
| 1001 | 30000 | 2010-01-01, UNTIL_CHANGED |
| 1002 | 25000 | 1995-01-01, 2010-01-01    |

Here is the query.

```
VALIDTIME AS OF DATE'2000-01-01'
SELECT E.EName as Name, S.Sal as Salary
FROM Employee E, Salary S
WHERE E.E_Id = S.E_Id;
```

It produces this data as the result.

| Name  | Salary |
|-------|--------|
| Sania | 10000  |
| Ash   | 25000  |

### Example 3: Select Transaction-Time Rows for a Specific Date and Time

This example extracts stock details as of a specific timestamp.

Data from this stock table contains a transaction-time dimension: the TransactionTime data column, RecordedTime.

---

| StockName | StockValue | RecordedTime                                   |
|-----------|------------|------------------------------------------------|
| Teradata  | 38         | 2006-01-01 10:00:00.000000+00:00,              |
|           |            | 2006-01-01 12:10:10.000000+00:00               |
| Teradata  | 37         | 2006-01-01 12:10:10.000000+00:00,              |
|           |            | 2006-01-03 10:00:00.000000+00:00               |
| Teradata  | 40         | 2006-01-03 10:00:00.000000+00:00, UNTIL_CLOSED |

Here is the query.

```
TRANSACTIONTIME AS OF TIMESTAMP'2006-01-02 12:10:10.000000+00:00'
SELECT * FROM Stock;
```

It produces this data as the result.

| StockName | StockValue |
|-----------|------------|
| Teradata  | 37         |

## TENACITY= Data Set Option

Specifies how many hours that FastExport, FastLoad, or MultiLoad continues to try to log on again to Teradata if the maximum number of Teradata utilities are already running.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when creating and appending to DBMS tables using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                     |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                   |
| Defaults:    | 0 (FastLoad)<br>4 (FastExport, MultiLoad)                                                                                                                                                                                                                                                                                                                                                                      |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                       |
| Tip:         | The data set option has precedence over the LIBNAME option.                                                                                                                                                                                                                                                                                                                                                    |
| See:         | <a href="#">DBSLICEPARM= LIBNAME option</a> , <a href="#">DBSLICEPARM= data set option</a> , <a href="#">DBSLICEPARM= system option</a> , <a href="#">FASTEEXPORT= LIBNAME option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">SLEEP= LIBNAME option</a> , <a href="#">SLEEP= data set option</a> , <a href="#">Using the TPT API</a> |

## Syntax

**TENACITY=***number-of-hours*

### Syntax Description

#### *number-of-hours*

specifies the number of hours to continue to try again to log on to Teradata.

## Details

Use this option to indicate to FastExport, FastLoad, or MultiLoad how long to continue retrying a logon operation when the maximum number of utilities are already running. (The maximum number of Teradata utilities that can run concurrently varies from 5 to 15, depending on the database server value.) The default value for TENACITY= is four hours. The value specified for TENACITY= must be greater than zero.

Use TENACITY= with [SLEEP=](#), which specifies the number of minutes that FastExport, FastLoad, or MultiLoad waits before it retries logging on to Teradata. SLEEP= and TENACITY= function very much like the SLEEP and TENACITY run-time options of the native Teradata FastExport, FastLoad, or MultiLoad utility.

Here is an example of the message that is written to the SAS log if the time period that TENACITY= specifies is exceeded.

```
ERROR: MultiLoad failed unexpectedly with returncode 12
```

Check the FastExport, FastLoad, or MultiLoad log for more information about the cause of the FastExport, FastLoad, or MultiLoad failure. SAS does not receive any informational messages from Teradata in either of these situations:

- when the currently run FastExport, FastLoad, or MultiLoad process waits because the maximum number of utilities are already running
- if FastExport, FastLoad, or MultiLoad is terminated because the time limit that TENACITY= specifies has been exceeded

The native Teradata FastExport, FastLoad, or MultiLoad utility sends messages associated with SLEEP= and TENACITY= only to the FastExport, FastLoad, or MultiLoad log. So nothing is written to the SAS log.

## TPT= Data Set Option

Specifies whether SAS uses the TPT API to load data for Fastload, MultiLoad, or Multi-Statement insert requests.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Default:     | YES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">LOGDB= LIBNAME option</a> , <a href="#">“Maximizing Teradata Load and Read Performance”</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">MULTISTMT= data set option</a> , <a href="#">TPT= LIBNAME option</a> , <a href="#">TPT_APPL_PHASE= data set option</a> , <a href="#">TPT_BUFFER_SIZE= data set option</a> , <a href="#">TPT_CHECKPOINT= data set option</a> , <a href="#">TPT_DATA_ENCRYPTION= data set option</a> , <a href="#">TPT_ERROR_TABLE_1= data set option</a> , <a href="#">TPT_ERROR_TABLE_2= data set option</a> , <a href="#">TPT_LOG_TABLE= data set option</a> , <a href="#">TPT_MAX_SESSIONS= LIBNAME option</a> , <a href="#">TPT_MAX_SESSIONS= data set option</a> , <a href="#">TPT_MIN_SESSIONS= data set option</a> , <a href="#">TPT_PACK= data set option</a> , <a href="#">TPT_PACKMAXIMUM= data set option</a> , <a href="#">TPT_RESTART= data set option</a> , <a href="#">TPT_TRACE_LEVEL= data set option</a> , <a href="#">TPT_TRACE_LEVEL_INF= data set option</a> , <a href="#">TPT_TRACE_OUTPUT= data set option</a> , <a href="#">TPT_WORK_TABLE= data set option</a> , <a href="#">Maximizing Teradata Load Performance</a> , <a href="#">“Using the TPT API”</a> |

## Syntax

**TPT=YES | NO**

### Syntax Description

#### YES

specifies that SAS uses the TPT API when Fastload, MultiLoad, or Multi-Statement insert is requested.

#### NO

specifies that SAS does not use the TPT API when Fastload, MultiLoad, or Multi-Statement insert is requested.

## Details

By using the TPT API, you can load data into a Teradata table without working directly with such stand-alone Teradata utilities as Fastload, MultiLoad, or TPump. When TPT=YES (the default), SAS uses the TPT API load driver for FastLoad, the update driver for MultiLoad, and the stream driver for Multi-Statement insert.

When TPT=YES, sometimes SAS cannot use the TPT API due to an error or because it is not installed on the system. When this happens, SAS does not produce an error, but it still tries to load data using the requested load method (Fastload, MultiLoad, or Multi-Statement insert). To check whether SAS used the TPT API to load data, look for a similar message to this one in the SAS log:

*NOTE: Teradata connection: TPT FastLoad/MultiLoad/MultiStatement insert has read n row(s).*

## Example: Load Data Using the TPT API

In this example, SAS data is loaded into Teradata using the TPT API. This is the default method of loading when Fastload, MultiLoad, or Multi-Statement insert is requested. SAS still tries to load data even if it cannot use the TPT API.

```
libname tera Teradata user=myusrl pw=mypwd1;
/* Create data */
data testdata;
do i=1 to 100;
    output;
end;
run;
/* Load using FastLoad TPT. This note appears in the SAS log if SAS uses TPT.
NOTE: Teradata connection: TPT FastLoad has inserted 100 row(s).*/
data tera.testdata(FASTLOAD=YES TPT=YES);
set testdata;
run;
```

## TPT\_APPL\_PHASE= Data Set Option

Specifies whether a load process that is being restarted has failed in the application phase.

Valid in: PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Bulk Loading

Default: NO

Requirement: To use this option, you must first specify TPT=YES.

Data source: Teradata

See: [Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, MULTILOAD= data set option, TPT= LIBNAME option, TPT= data set option, TPT\\_CHECKPOINT\\_DATA= data set option, TPT\\_RESTART= data set option](#)

## Syntax

**TPT\_APPL\_PHASE=YES | NO**

### Syntax Description

#### **YES**

specifies that the Fastload or MultiLoad run that is being restarted has failed in the application phase. This is valid only when SAS uses the TPT API.

#### **NO**

specifies that the load process that is being restarted has not failed in the application phase.

---

## Details

SAS can restart from checkpoints any Fastload, MultiLoad, and Multi-Statement insert that is run using the TPT API. The restart procedure varies: It depends on whether checkpoints were recorded and in which phase the step failed during the load process. Teradata loads data in two phases: the acquisition phase and the application phase. In the acquisition phase, data transfers from SAS to Teradata. After this phase, SAS has no more data to transfer to Teradata. If failure occurs after this phase, set TPT\_APPL\_PHASE=YES in the restart step to indicate that restart is in the application phase. (Multi-Statement insert does not have an application phase. Therefore, it does not need to be restarted if it fails after the acquisition phase.)

Use OBS=1 for the source data set when restart occurs in the application phase. When SAS encounters TPT\_RESTART=YES and TPT\_APPL\_PHASE=YES, it initiates restart in the application phase. No data from the source data set is actually sent. If you use OBS=1 for the source data set, the SAS step completes as soon as it reads the first record. (It actually throws away the record because SAS already sent all data to Teradata for loading.)

---

## Example: Restart after Failure

Here is a sample SAS program that failed after the acquisition phase.

```
libname x teradata user=mysrv1 pw=mypwd1;
data x.test(MULTILOAD=YES TPT=YES CHECKPOINT=7);
do i=1 to 20;
output;
end;
run;
ERROR: Teradata connection: Failure occurred after the acquisition phase.
Restart outside of SAS using checkpoint data 14.
```

Set TPT\_APPL\_PHASE=YES to restart when failure occurs in the application phase because SAS has already sent all data to Teradata.

```
proc append base=x.test (MULTILOAD=YES TPT_RESTART=YES
```

```
TPT_CHECKPOINT_DATA=14 TPT_APPL_PHASE=YES) data=test(obs=1);
run;
```

## TPT\_BLOCK\_SIZE= Data Set Option

Specifies the block size in bytes for TPT Export.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Bulk Loading                                                             |
| Default:     | Varies (see Details)                                                     |
| Range:       | Varies (see Details)                                                     |
| Data source: | Teradata                                                                 |
| Note:        | Support for this data set option was added in SAS Viya 3.5.              |

## Syntax

**TPT\_BLOCK\_SIZE=***integer*

### Required Argument

*integer*

specifies the block size in bytes when running TPT Export.

## Details

Using this option can improve performance when transferring data.

The value that you specify cannot be larger than the message size that is supported by your Teradata DBMS. If the value that you specify is too large, SAS/ACCESS automatically resets the value to the maximum that is allowed for a block size and the job continues.

The default value for TPT\_BLOCK\_SIZE= varies, based on your Teradata version.

*Table 13.8 TPT\_BLOCK\_SIZE= Default and Valid Values*

| Version                    | Default  | Range        |
|----------------------------|----------|--------------|
| Teradata 15.10 and earlier | 64330    | 256–64330    |
| Teradata 16.00 and later   | 16775168 | 256–16775168 |

---

## TPT\_BUFFER\_SIZE= Data Set Option

Specifies the output buffer size in kilobytes when SAS sends data to Teradata with Fastload or MultiLoad using the TPT API.

|              |                                                                                                                                                                         |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                |
| Category:    | Bulk Loading                                                                                                                                                            |
| Default:     | 64                                                                                                                                                                      |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                     |
| Data source: | Teradata                                                                                                                                                                |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, TPT= LIBNAME option, TPT= data set option</a> |

---

## Syntax

**TPT\_BUFFER\_SIZE=integer**

### Syntax Description

**integer**

specifies the size of data parcels in kilobytes from 1 through 64.

---

## Details

You can use the output buffer size to control the amount of data that is transferred in each parcel from SAS to Teradata when using the TPT API. A larger buffer size can reduce processing overhead by including more data in each parcel. See your Teradata documentation for details.

---

## TPT\_CHECKPOINT\_DATA= Data Set Option

Specifies the checkpoint data to return to Teradata when restarting a failed MultiLoad or Multi-Statement step that uses the TPT API.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Bulk Loading                                                             |
| Default:     | none                                                                     |
| Requirement: | To use this option, you must first specify TPT=YES and TPT_RESTART=YES.  |
| Data source: | Teradata                                                                 |

See: [BULKLOAD=](#) data set option, “Maximizing Teradata Load and Read Performance”, [MULTILOAD=](#) data set option, [MULTISTMT=](#) data set option, [TPT= LIBNAME option](#), [TPT\\_APPL\\_PHASE=](#) data set option, [TPT\\_RESTART=](#) data set option, “Using the TPT API”

---

## Syntax

**TPT\_CHECKPOINT\_DATA**=*checkpoint\_data\_in\_error\_message*

### Syntax Description

***checkpoint\_data\_in\_error\_message***

specifies the value to use to restart a failed MultiLoad or Multi-Statement step that uses the TPT API.

---

## Details

SAS can restart from the last checkpoint any failed Fastload, MultiLoad, and Multi-Statement insert that are run using the TPT API. Teradata returns a checkpoint value each time MultiLoad or Multi-Statement records a checkpoint. The SAS log contains this value when a load fails. SAS must provide the same value as a data set option when it tries to restart the load process.

Here are the rules that govern restart.

- The TPT API does not return a checkpoint value when FastLoad records a checkpoint. Therefore, you do not need to specify **TPT\_CHECKPOINT\_VALUE**= when you use FastLoad. Specify **TPT\_RESTART**= instead.
- If the default error table name, work table name, or restart table name is overridden, then SAS must use the same name while restarting the load process.
- Teradata loads data in two phases: the acquisition phase and the application phase. In the acquisition phase, data transfers from SAS to Teradata. After this phase, SAS has no more data to transfer to Teradata. If failure occurs after this phase, set **TPT\_APPL\_PHASE=YES** while restarting. (Multi-Statement insert does not have an application phase. Therefore, it does not need to be restarted if it fails after the acquisition phase.) Use **OBS=1** for the source data set because SAS has already sent the data to Teradata. Therefore, there is no need to send any more data.
- If failure occurred before the acquisition phase ended and the load process recorded no checkpoints, you must restart the load process from the beginning by specifying **TPT\_RESTART=YES**. However, you do not need to specify **TPT\_CHECKPOINT\_VALUE**= because no checkpoints were recorded. The error message in the SAS log provides all needed information for restart.

## Examples

### Example 1

In this example, assume that the MultiLoad step that uses the TPT API fails before the acquisition phase ends and no options were set to record checkpoints.

```
libname x teradata user=myusr1 pw=mypwd1;
data test;In
do i=1 to 100;
output;
end;
run;
/* Set TPT=YES is optional because it is the default. */
data x.test(MULTILOAD=YES TPT=YES);
set test;
run;
```

This error message is sent to the SAS log. You do not need to specify TPT\_CHECKPOINT\_DATA= because no checkpoints were recorded.

```
ERROR: Teradata connection: Correct error and restart as an APPEND
process with option TPT_RESTART=YES. Since no checkpoints were taken,
if the previous run used FIRSTOBS=n, use the same value in the restart.
```

### Example 2

Here is an example of the restart step.

```
proc append data=test base=x.test(FASTLOAD=YES TPT=YES TPT_RESTART=YES);
run;
```

### Example 3

In this next example, failure occurs after checkpoints are recorded.

```
libname tera teradata user=myusr1 pw=mypwd1;
/* Create data */
data testdata;
do i=1 to 100;
output;
end;
run;
/* Assume that this step fails after loading row 19. */
data x.test(MULTISTMT=YES CHECKPOINT=3);
set testdata;
run;
```

Here is the resulting error when it fails after loading 18 rows.

```
ERROR: Teradata connection: Correct error and restart as
```

an APPEND process with option TPT\_RESTART=YES. If the previous run used FIRSTOBS=n, use the value ( n-1+ 19 ) for FIRSTOBS in the restart. Otherwise use FIRSTOBS= 19. Also specify TPT\_CHECKPOINT\_DATA= 18.

You can restart the failed step with this code.

```
proc append base=x.test (MULTISTMT=YES TPT_RESTART=YES
                        TPT_CHECKPOINT_DATA=18) data=test(firstobs=19);
run;
```

If failure occurs after the end of the acquisition phase, you must write a custom C++ program to restart from the point where it stopped.

## Example 4

Here is a sample SAS program that failed after the acquisition phase and the resulting error message.

```
libname x teradata user=myusr1 pw=mypwd1;
data x.test (MULTILOAD=YES TPT=YES CHECKPOINT=7);
do i=1 to 20;
output;
end;
run;
ERROR: Teradata connection: Failure occurred after the acquisition phase.
Restart outside of SAS using checkpoint data 14.
```

Set TPT\_APPL\_PHASE=YES to restart when failure occurs in the application phase because SAS has already sent all data to Teradata.

```
proc append base=x.test (MULTILOAD=YES TPT_RESTART=YES
                        TPT_CHECKPOINT_DATA=14 TPT_APPL_PHASE=YES) data=test(obs=1);
run;
```

## TPT\_DATA\_ENCRYPTION= Data Set Option

Specifies whether to fully encrypt SQL requests, responses, and data when SAS sends data to Teradata for Fastload, MultiLoad, or Multi-Statement insert that uses the TPT API.

Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)

Category: Bulk Loading

Default: NO

Requirement: To use this option, you must first specify TPT=YES.

Data source: Teradata

See: [Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, MULTILOAD= data set option, MULTISTMT= data set option, TPT= LIBNAME option, TPT= data set option](#)

## Syntax

**TPT\_DATA\_ENCRYPTION=YES | NO**

### Syntax Description

#### YES

specifies that all communication between the Teradata client and server is encrypted when using the TPT API.

#### NO

specifies that all communication between the Teradata client and server is not encrypted when using the TPT API.

---

## Details

You can ensure that SQL requests, responses, and data that are transferred between the Teradata client and server are encrypted when using the TPT API. See your Teradata documentation for details.

---

## TPT\_ERROR\_TABLE\_1= Data Set Option

Specifies the name of the first error table for SAS to use when using the TPT API with Fastload or MultiLoad.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                         |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Default:     | table_name_ET                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                                                                                                                                                                                |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                                                                                                                                                                              |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| See:         | <a href="#">Maximizing Teradata Load Performance</a> , “Using the TPT API”, <a href="#">BULKLOAD= LIBNAME</a> option, <a href="#">BULKLOAD= data set option</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">TPT= LIBNAME</a> option, <a href="#">TPT= data set option</a> , <a href="#">TPT_ERROR_TABLE_2= data set option</a> , <a href="#">TPT_LOG_TABLE= data set option</a> , <a href="#">TPT_WORK_TABLE= data set option</a> |

---

## Syntax

**TPT\_ERROR\_TABLE\_1=valid\_teradata\_table\_name**

## Syntax Description

### ***valid\_teradata\_table\_name***

specifies the name of the first error table for SAS to use when using the TPT API to load data with Fastload or MultiLoad.

---

## Details

Fastload and MultiLoad require an error table to hold records that were rejected during the acquisition phase. If you do not specify an error table, Teradata appends "\_ET" to the name of the target table to load and uses it as the first error table by default. You can override this name by specifying **TPT\_ERROR\_TABLE\_1=**. If you do this and the load step fails, you must specify the same name when restarting. For information about errors that are logged in this table, see your Teradata documentation.

The name that you specify in **TPT\_ERROR\_TABLE\_1=** must be unique. It cannot be the name of an existing table unless it is in a restart scenario.

---

## Example: Specify Different Names for Two Error Tables

In this example, a different name is provided for both the first and second error tables that Fastload and MultiLoad use with the TPT API.

```
libname tera teradata user=myusr1 pw=mypwd1;
/* Load using Fastload TPT. Use alternate names for the error tables. */
data tera.testdata(FASTLOAD=YES TPT_ERROR_TABLE_1=testerror1
                   TPT_ERROR_TABLE_2=testerror2);
  i=1;output; i=2;output;
run;
```

---

## TPT\_ERROR\_TABLE\_2= Data Set Option

Specifies the name of the second error table for SAS to use when using the TPT API with Fastload or MultiLoad.

|              |                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                               |
| Category:    | Bulk Loading                                                                                                                                                           |
| Default:     | table_name_UV                                                                                                                                                          |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                      |
| Requirement: | To use this option, you must first specify <b>TPT=YES</b> .                                                                                                            |
| Data source: | Teradata                                                                                                                                                               |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, MULTILOAD= data set option, TPT= LIBNAME</a> |

option, TPT= data set option, TPT\_ERROR\_TABLE\_1= data set option,  
TPT\_LOG\_TABLE= data set option, TPT\_WORK\_TABLE= data set option

---

## Syntax

**TPT\_ERROR\_TABLE\_2=valid\_teradata\_table\_name**

### Syntax Description

#### ***valid\_teradata\_table\_name***

specifies the name of the second error table for SAS to use when using the TPT API to load data with Fastload or MultiLoad.

---

## Details

Fastload and MultiLoad require an error table to hold records that were rejected during the acquisition phase. If you do not specify an error table, Teradata appends "\_UV" to the name of the target table to load and uses it as the second error table by default. You can override this name by setting TPT\_ERROR\_TABLE\_2=. If you do this and the load step fails, you must specify the same name when restarting. For information about errors that are logged in this table, see your Teradata documentation.

The name that you specify in TPT\_ERROR\_TABLE\_2= must be unique. It cannot be the name of an existing table unless it is in a restart scenario.

---

## Example: Specify Different Names for First and Second Error Tables

In this example, a different name is provided for both the first and second error tables that Fastload and MultiLoad use with the TPT API.

```
libname tera teradata user=myusr1 pw=mypwd1;
/* Load using Fastload TPT. Use alternate names for the error tables. */
data tera.testdata(FASTLOAD=YES TPT_ERROR_TABLE_1=testerror1
                   TPT_ERROR_TABLE_2=testerror2);
  i=1;output; i=2;output;
run;
```

---

## TPT\_LOG\_TABLE= Data Set Option

Specifies the name of the restart log table for SAS to use when using the TPT API with Fastload, MultiLoad, or Multi-Statement insert.

|              |                                                                                                                                                                                                                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                 |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                             |
| Default:     | table_name_RS                                                                                                                                                                                                                                                                                                                            |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                                                                        |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                                                                      |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                 |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, MULTILOAD= data set option, MULTISTMT= data set option, TPT= LIBNAME option, TPT= data set option, TPT_ERROR_TABLE_1= data set option, TPT_ERROR_TABLE_2= data set option, TPT_WORK_TABLE= data set option</a> |

## Syntax

**TPT\_LOG\_TABLE=valid\_teradata\_table\_name**

### Syntax Description

***valid\_teradata\_table\_name***

specifies the name of the restart log table for SAS to use when using the TPT API to load data with Fastload or MultiLoad.

## Details

To use this option, you must first specify TPT=YES. This option is valid only when using the TPT API.

Fastload, MultiLoad, and Multi-Statement insert that use the TPT API require a restart log table. If you do not specify a restart log table, Teradata appends "\_RS" to the name of the target load table and uses it as the restart log table by default. You can override this name by specifying TPT\_LOG\_TABLE=. If you do this and the load step fails, you must specify the same name when restarting.

The name that you specify in TPT\_LOG\_TABLE= must be unique. It cannot be the name of an existing table unless it is in a restart scenario.

## Example: Specify a Different Name for the Restart Log Table

In this example, a different name is provided for the restart log table that Multi-Statement uses with the TPT API.

```
libname tera teradata user=myusr1 pw=mypwd1;
/* Load using Fastload TPT. Use alternate names for the log table. */
data tera.testdata(MULTISTMT=YES TPT_LOG_TABLE=restarttab);
```

```
i=1;output; i=2;output;  
run;
```

---

## TPT\_MAX\_SESSIONS= Data Set Option

Specifies the maximum number of sessions for Teradata to use when using the TPT API with FastLoad, MultiLoad, or Multi-Statement insert.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                  |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Default:     | 4                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                                                                                                                                                                         |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                                                                                                                                                                       |
| Interaction: | [precedence] 1. TPT_MAX_SESSIONS= data set option, 2. TPT_MAX_SESSIONS= LIBNAME option, 3. SAS_TPT_MAX_SESSIONS environment variable, 4. the default value                                                                                                                                                                                                                                                                                |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Note:        | The default value was changed to 4 for SAS 9.4.                                                                                                                                                                                                                                                                                                                                                                                           |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">MULTISTMT= data set option</a> , <a href="#">TPT= LIBNAME option</a> , <a href="#">TPT= data set option</a> , <a href="#">TPT_MAX_SESSIONS= LIBNAME option</a> , <a href="#">TPT_MIN_SESSIONS= data set option</a> , “Maximizing Teradata Load and Read Performance”, “Using the TPT API” |

---

## Syntax

**TPT\_MAX\_SESSIONS=***integer*

### Syntax Description

#### *integer*

specifies the maximum number of sessions for Teradata to use when using the TPT API to load data with FastLoad, MultiLoad, or Multi-Statement insert.

---

## Details

You can control the number of sessions for Teradata to use when using the TPT API to load data with MultiLoad. The maximum value cannot be more than the number of available Access Module Processors (AMPs). See your Teradata documentation for details.

---

## TPT\_MIN\_SESSIONS= Data Set Option

Specifies the minimum number of sessions for Teradata to use when using the TPT API with FastLoad, MultiLoad, or Multi-Statement insert.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Default:     | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Interaction: | [precedence] 1. TPT_MIN_SESSIONS= data set option, 2. TPT_MIN_SESSIONS= LIBNAME option, 3. SAS_TPT_MIN_SESSIONS environment variable, 4. the default value                                                                                                                                                                                                                                                                                                                                       |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| See:         | <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">MULTISTMT= data set option</a> , <a href="#">TPT= LIBNAME option</a> , <a href="#">TPT= data set option</a> , <a href="#">TPT_MAX_SESSIONS= LIBNAME option</a> , <a href="#">TPT_MAX_SESSIONS= data set option</a> , <a href="#">TPT_MIN_SESSIONS= data set option</a> ,<br>“Maximizing Teradata Load and Read Performance”, “Using the TPT API” |

---

## Syntax

**TPT\_MIN\_SESSIONS=***integer*

### Syntax Description

***integer***

specifies the minimum number of sessions for Teradata to use when using the TPT API to load data with FastLoad, MultiLoad, or Multi-Statement insert.

---

## Details

You can control the number of sessions that are required before using the TPT API to load data with MultiLoad. This value must be greater than zero and less than the maximum number of required sessions. See your Teradata documentation for details.

---

## TPT\_PACK= Data Set Option

Specifies the number of statements to pack into a Multi-Statement insert request when using the TPT API.

|               |                                                                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                         |
| Category:     | Bulk Loading                                                                                                                                                                     |
| Default:      | 20                                                                                                                                                                               |
| Restrictions: | The maximum value is 600. See your Teradata documentation for details.<br>This option is valid only when using the TPT API.                                                      |
| Requirement:  | To use this option, you must first specify TPT=YES.                                                                                                                              |
| Data source:  | Teradata                                                                                                                                                                         |
| See:          | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, MULTISTMT= data set option, TPT= LIBNAME option, TPT= data set option, TPT_PACKMAXIMUM= data set option</a> |

## Syntax

**TPT\_PACK=***integer*

### Syntax Description

#### *integer*

specifies the number of statements to pack into a Multi-Statement insert request when using the TPT API.

## TPT\_PACKMAXIMUM= Data Set Option

Specifies whether to pack the maximum possible or default number of statements into Multi-Statement insert requests when using the TPT API.

|              |                                                                                                                                                                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                             |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                         |
| Default:     | NO                                                                                                                                                                                                                                                                                   |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                    |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                  |
| Data source: | Teradata                                                                                                                                                                                                                                                                             |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, MULTISTMT= data set option, TPT= LIBNAME option, TPT= data set option, TPT_PACK= data set option, TPT_TRACE_LEVEL= data set option, TPT_TRACE_LEVEL_INF= data set option, TPT_TRACE_OUTPUT= data set option</a> |

## Syntax

**TPT\_PACKMAXIMUM=**YES | NO

## Syntax Description

### **YES**

specifies that the maximum possible number of statements be packed into Multi-Statement insert requests when using the TPT API.

### **NO**

specifies that the default number of statements be packed into Multi-Statement insert requests when using the TPT API.

## Details

When TPT\_PACKMAXIMUM=YES, the maximum possible number of statements that can fit in a Multi-Statement request is determined dynamically. See your Teradata documentation for details.

## TPT\_RESTART= Data Set Option

Specifies that a failed FastLoad, MultiLoad, or Multi-Statement run that used the TPT API is being restarted.

|              |                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                               |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                                                                                                                                  |
| Default:     | NO                                                                                                                                                                                                                                                                                                                                                                                                            |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                                                                                                                                             |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                                                                                                                                           |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                                                                                      |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API</a> , <a href="#">BULKLOAD= LIBNAME option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">MULTILOAD= data set option</a> , <a href="#">MULTISTMT= data set option</a> , <a href="#">BULKLOAD= data set option</a> , <a href="#">TPT_APPL_PHASE= data set option</a> , <a href="#">TPT_CHECKPOINT_DATA= data set option</a> |

## Syntax

**TPT\_RESTART=YES | NO**

## Syntax Description

### **YES**

specifies that the load process is being restarted.

### **NO**

specifies that the load process is not being restarted.

## Details

SAS can restart from checkpoints any FastLoad, MultiLoad, and Multi-Statement insert that are run using the TPT API. The restart procedure varies: It depends on whether checkpoints were recorded and in which phase the step failed during the load process. The error message in the log is extremely important and contains instructions on how to restart.

Here are the rules that govern restart.

- The TPT API does not return a checkpoint value when FastLoad records a checkpoint. Therefore, you do not need to specify TPT\_CHECKPOINT\_VALUE= when you use FastLoad. Specify TPT\_RESTART= instead.
- If the default error table name, work table name, or restart table name is overridden, then SAS must use the same name while restarting the load process.
- Teradata loads data in two phases: the acquisition phase and the application phase. In the acquisition phase, data transfers from SAS to Teradata. After this phase, SAS has no more data to transfer to Teradata. If failure occurs after this phase, set TPT\_APPL\_PHASE=YES while restarting. (Multi-Statement insert does not have an application phase. Therefore, it does not need to be restarted if it fails after the acquisition phase.) Use OBS=1 for the source data set because SAS has already sent the data to Teradata. Therefore, there is no need to send any more data.
- If failure occurred before the acquisition phase ended and the load process recorded no checkpoints, you must restart the load process from the beginning by specifying TPT\_RESTART=YES. However, you do not need to specify TPT\_CHECKPOINT\_VALUE= because no checkpoints were recorded. The error message in the SAS log provides all needed information for restart.

---

## Examples

---

### Example 1

In this example, assume that the MultiLoad step that uses the TPT API fails before the acquisition phase ends and no options were set to record checkpoints.

```
libname x teradata user=myusr1 pw=mypwd1;
data test;In
do i=1 to 100;
output;
end;
run;
/* Set TPT=YES is optional because it is the default. */
data x.test (MULTILOAD=YES TPT=YES);
set test;
run;
```

This error message is sent to the SAS log. You do not need to specify TPT\_CHECKPOINT\_DATA= because no checkpoints were recorded.

ERROR: Teradata connection: Correct error and restart as an APPEND process with option TPT\_RESTART=YES. Since no checkpoints were taken, if the previous run used FIRSTOBS=n, use the same value in the restart.

---

## Example 2

Here is an example of the restart step.

```
proc append data=test base=x.test (MULTILOAD=YES
    TPT=YES TPT_RESTART=YES) ;
run;
```

---

## Example 3

In this next example, failure occurs after checkpoints are recorded.

```
libname tera teradata user=myusr1 pw=mypwd1;
/* Create data */
data testdata;
do i=1 to 100;
    output;
end;
run;
/* Assume that this step fails after loading row 19. */
data x.test (MULTISTMT=YES CHECKPOINT=3) ;
set testdata;
run;
```

Here is the resulting error when it fails after loading 18 rows.

ERROR: Teradata connection: Correct error and restart as an APPEND process with option TPT\_RESTART=YES. If the previous run used FIRSTOBS=n, use the value ( n-1+ 19) for FIRSTOBS in the restart. Otherwise use FIRSTOBS=19. Also specify TPT\_CHECKPOINT\_DATA= 18.

You can restart the failed step with this code.

```
proc append base=x.test (MULTISTMT=YES TPT_RESTART=YES
    TPT_CHECKPOINT_DATA=18) data=test(firstobs=19) ;
run;
```

If failure occurs after the end of the acquisition phase, you must write a custom C++ program to restart from the point where it stopped.

---

## Example 4

Here is a sample SAS program that failed after the acquisition phase and the resulting error message.

```
libname x teradata user=myusr1 pw=mypwd1;
data x.test (MULTILOAD=YES TPT=YES CHECKPOINT=7) ;
do i=1 to 20;
```

```

output;
end;
run;
ERROR: Teradata connection: Failure occurred after the acquisition
phase. Restart outside of SAS using checkpoint data 14.

```

Set TPT\_APPL\_PHASE=YES to restart when failure occurs in the application phase because SAS has already sent all data to Teradata.

```

proc append base=x.test (MULTILOAD=YES TPT_RESTART=YES
                        TPT_CHECKPOINT_DATA=14 TPT_APPL_PHASE=YES) data=test(obs=1);
run;

```

You must always use TPT\_CHECKPOINT\_DATA= with TPT\_RESTART= for MultLoad and Multi-Statement insert.

## TPT\_TRACE\_LEVEL= Data Set Option

Specifies the required tracing level for sending data to Teradata and using the TPT API with Fastload, MultiLoad, or Multi-Statement insert.

|              |                                                                                                                                                                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                 |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                             |
| Default:     | 1                                                                                                                                                                                                                                                                                                        |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                                        |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                                      |
| Data source: | Teradata                                                                                                                                                                                                                                                                                                 |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, MULTILOAD= data set option, MULTISTMT= data set option, TPT= LIBNAME option, TPT= data set option, TPT_TRACE_LEVEL_INF= data set option, TPT_TRACE_OUTPUT= data set option</a> |

## Syntax

**TPT\_TRACE\_LEVEL=integer**

### Syntax Description

#### **integer**

specifies the needed tracing level (1 to 9) when loading data to Teradata.

*Table 13.9 Teradata Tracing Levels*

| Tracing Level | Description                  |
|---------------|------------------------------|
| 1             | no tracing                   |
| 2             | operator-level general trace |

| Tracing Level | Description                                       |
|---------------|---------------------------------------------------|
| 3             | operator-level command-line interface (CLI) trace |
| 4             | operator-level notify method trace                |
| 5             | operator-level common library trace               |
| 6             | all operator-level traces                         |
| 7             | Telnet API (TELAPI) layer general trace           |
| 8             | PutRow or GetRow trace                            |
| 9             | operator log message information                  |

## Details

You can perform debugging by writing diagnostic messages to an external log file when loading data to Teradata using the TPT API. If you do not specify a name in TPT\_TRACE\_OUTPUT= for the log file, a default name is generated using the current timestamp. See your Teradata documentation for details.

---

## TPT\_TRACE\_LEVEL\_INF= Data Set Option

Specifies the tracing level for the required infrastructure for sending data to Teradata and using the TPT API with Fastload, MultiLoad, or Multi-Statement insert.

|              |                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                             |
| Category:    | Bulk Loading                                                                                                                                                                                                                                                                                         |
| Default:     | 1                                                                                                                                                                                                                                                                                                    |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                                                                    |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                                                                  |
| Data source: | Teradata                                                                                                                                                                                                                                                                                             |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, MULTILOAD= data set option, MULTISTMT= data set option, TPT= LIBNAME option, TPT= data set option, TPT_TRACE_LEVEL= data set option, TPT_TRACE_OUTPUT= data set option</a> |

---

## Syntax

**TPT\_TRACE\_LEVEL\_INF=integer**

## Syntax Description

### **integer**

specifies the needed infrastructure tracing level (10 to 18) when loading data to Teradata.

*Table 13.10 Infrastructure Tracing Levels*

| Tracing Level | Description                                       |
|---------------|---------------------------------------------------|
| 10            | no tracing                                        |
| 11            | operator-level general trace                      |
| 12            | operator-level command-line interface (CLI) trace |
| 13            | operator-level notify method trace                |
| 14            | operator-level common library trace               |
| 15            | all operator-level traces                         |
| 16            | Telnet API (TELAPI) layer general trace           |
| 17            | PutRow or GetRow trace                            |
| 18            | operator log message information                  |

---

## Details

You can perform debugging by writing diagnostic messages to an external log file when loading data to Teradata using the TPT API. If you do not specify a name in PT\_TRACE\_OUTPUT= for the log file, a default name is generated using the current timestamp. See your Teradata documentation for details.

---

## TPT\_TRACE\_OUTPUT= Data Set Option

Specifies the name of the external file for SAS to use for tracing when using the TPT API with Fastload, MultiLoad, or Multi-Statement insert.

- Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)
- Category: Bulk Loading
- Default: *driver\_name timestamp*
- Restriction: This option is valid only when using the TPT API.
- Requirement: To use this option, you must first specify TPT=YES.

|              |                                                                                                                                                                                                                                                                                                                                           |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | Teradata                                                                                                                                                                                                                                                                                                                                  |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, BULKLOAD= LIBNAME option, BULKLOAD= data set option, MULTILOAD= data set option, MULTISTMT= data set option, TPT= LIBNAME option, TPT= data set option, TPT_TRACE_LEVEL= data set option, TPT_TRACE_LEVEL_INF= data set option, TPT_PACKMAXIMUM= data set option</a> |

## Syntax

**TPT\_TRACE\_OUTPUT=*file-name***

### Syntax Description

***file-name***

specifies the name of the external file to use for tracing. The name must be a valid file name for the operating system.

## Details

You can save diagnostic messages to an external log file when loading data to Teradata using the TPT API. If you request tracing but specify no name in TPT\_TRACE\_OUTPUT= for the log file, a default name is generated using the name of the driver and the current timestamp. Otherwise, the name that you specify is used for tracing messages. If the file already exists, it is overwritten. See your Teradata documentation for details.

## TPT\_WORK\_TABLE= Data Set Option

Specifies the name of the work table for SAS to use when using the TPT API with MultiLoad.

|              |                                                                                                                                                                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                               |
| Category:    | Bulk Loading                                                                                                                                                                                                                                           |
| Default:     | table_name_WT                                                                                                                                                                                                                                          |
| Restriction: | This option is valid only when using the TPT API.                                                                                                                                                                                                      |
| Requirement: | To use this option, you must first specify TPT=YES.                                                                                                                                                                                                    |
| Data source: | Teradata                                                                                                                                                                                                                                               |
| See:         | <a href="#">Maximizing Teradata Load Performance, Using the TPT API, MULTILOAD= data set option, TPT= LIBNAME option, TPT= data set option, TPT_ERROR_TABLE_1= data set option, TPT_ERROR_TABLE_2= data set option, TPT_LOG_TABLE= data set option</a> |

## Syntax

**TPT\_WORK\_TABLE=***valid\_teradata\_table\_name*

### Syntax Description

***valid\_teradata\_table\_name***

specifies the name of the work table for SAS to use when using the TPT API to load data with MultiLoad.

## Details

MultiLoad inserts that use the TPT API require a work table. If you do not specify a work table, Teradata appends "\_WT" to the name of the target table to load and uses it as the work table by default. You can override this name by specifying TPT\_WORK\_TABLE=. If you do this and the load step fails, you must specify the same name when restarting.

The name that you specify in TPT\_WORK\_TABLE= must be unique. It cannot be the name of an existing table unless it is in a restart scenario.

## Example: Specify a Different Name for the Work Table

In this example, a different name is provided for the work table that MultiLoad uses with the TPT API.

```
libname tera teradata user=myusr1 pw=mypwd1;
/* Load using Multiload TPT. Use alternate names for the work table. */
data teratestdata(MULTILOAD=YES TPT_WORK_TABLE=worktab);
i=1;output; i=2;output;
run;
```

---

## TRANSCODE\_FAIL= Data Set Option

Lets you specify how to handle processing and notification of transcoding errors.

|               |                                                                                                                                                          |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:     | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                 |
| Category:     | Data Set Control                                                                                                                                         |
| Default:      | LIBNAME option value                                                                                                                                     |
| Restrictions: | This option is deprecated in SAS Viya 3.5. Use the SUB_CHAR= data set option instead.<br>This option is not available for use with SAS Embedded Process. |
| Data source:  | Hadoop, JDBC                                                                                                                                             |

- Note: Support for JDBC was added in SAS Viya 3.4.
- Tip: You can use TRANSCODE\_FAIL= to determine whether you want to halt or continue processing when transcoding errors are encountered.
- See: [SUB\\_CHAR= data set option](#), [TRANSCODE\\_FAIL= LIBNAME option](#)

## Syntax

**TRANSCODE\_FAIL=<ERROR> | <WARNING> | <SILENT>**

## Optional Arguments

### **ERROR**

stops processing data and provides an informative error message.

### **WARNING**

continues processing of data but provides an informative error message.

### **SILENT**

continues processing of data but suppresses messages.

## Details

The TRANSCODE\_FAIL= data set option is deprecated in the SAS Viya 3.5 (November 2019) release. Using this option stops processing and results in an error in the SAS log. Use the SUB\_CHAR= data set option instead.

## TRAP151= Data Set Option

Enables removal of columns that cannot be updated from a FOR UPDATE OF clause so that update of columns can proceed as normal.

- Valid in: DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)
- Category: Data Set Control
- Default: NO
- Data source: DB2 under z/OS

## Syntax

**TRAP151=YES | NO**

## Syntax Description

### YES

removes the non-updatable column that is designated in the error-151 and reprepares the statement for processing. This process is repeated until all columns that cannot be updated are removed, and all remaining columns can be updated.

### NO

disables TRAP151=. TRAP151= is disabled by default. It is not necessary to specify NO.

## Examples

### Example 1: SAS Log for TRAP151=YES

In this example, the SASTRACE=',,d' option is used so that you can see what occurs when TRAP151=YES:

```
proc fsedit data=x.v4(trap151=yes);
run;
SELECT * FROM V4 FOR FETCH ONLY
SELECT * FROM V4 FOR FETCH ONLY
SELECT "A", "X", "Y", "B", "Z", "C" FROM V4 FOR UPDATE OF "A", "X", "Y", "B", "Z", "C"
DB2 SQL Error, sqlca->sqlcode=-151
WARNING: SQLCODE -151: repreparing SELECT as:
      SELECT "A", "X", "Y", "B", "Z", "C" FROM V4 FOR UPDATE OF "A", "Y", "B", "Z", "C"
DB2 SQL Error, sqlca->sqlcode=-151
WARNING: SQLCODE -151: repreparing SELECT as:
      SELECT "A", "X", "Y", "B", "Z", "C" FROM V4 FOR UPDATE OF "A", "B", "Z", "C"
DB2 SQL Error, sqlca->sqlcode=-151
WARNING: SQLCODE -151: repreparing SELECT as:
      SELECT "A", "X", "Y", "B", "Z", "C" FROM V4 FOR UPDATE OF "A", "B", "C"
COMMIT WORK
NOTE: The PROCEDURE FSEDIT used 0.13 CPU seconds and 14367K.
```

### Example 2: SAS Log for TRAP151=NO

The next example features the same code with TRAP151 turned off:

```
proc fsedit data=x.v4(trap151=no);
run;
SELECT * FROM V4 FOR FETCH ONLY
SELECT * FROM V4 FOR FETCH ONLY
SELECT "A", "X", "Y", "B", "Z", "C" FROM V4 FOR UPDATE OF "A", "X", "Y", "B", "Z", "C"
DB2 SQL Error, sqlca->sqlcode=-151
ERROR: DB2 prepare error; DSNT4081 SQLCODE= -151, ERROR;
      THE UPDATE STATEMENT IS INVALID BECAUSE THE CATALOG DESCRIPTION OF COLUMN C
      INDICATES THAT IT CANNOT BE UPDATED.
COMMIT WORK
NOTE: The SAS System stopped processing this step because of errors.
NOTE: The PROCEDURE FSEDIT used 0.08 CPU seconds and 14367K.
```

## UPDATE\_ISOLATION\_LEVEL= Data Set Option

Specifies the degree of isolation of the current application process from other concurrently running application processes.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                                                                                                                                                                                                                                                                                                               |
| Category:    | Data Set Control                                                                                                                                                                                                                                                                                                                                                                                                       |
| Alias:       | UIL= [Greenplum, HAWQ, Microsoft SQL Server, SAP IQ]                                                                                                                                                                                                                                                                                                                                                                   |
| Default:     | LIBNAME option value                                                                                                                                                                                                                                                                                                                                                                                                   |
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, MySQL, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                                                  |
| Notes:       | Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                                                                                                                                      |
| See:         | <a href="#">READ_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= data set option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a> , <a href="#">UPDATE_LOCK_TYPE= data set option</a> , and DBMS-specific locking information in the reference section for your SAS/ACCESS interface |

## Syntax

**UPDATE\_ISOLATION\_LEVEL=DBMS-specific-value**

## Details

The degree of isolation indicates the degree to which these items are affected:

- Rows that the current application reads and updates are available to other concurrently executing applications.
- Update activity of other concurrently executing application processes can affect the current application.

## UPDATE\_LOCK\_TYPE= Data Set Option

Specifies how data in a DBMS table is locked during an update transaction.

|           |                                                                          |
|-----------|--------------------------------------------------------------------------|
| Valid in: | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category: | Data Access                                                              |
| Default:  | LIBNAME option value                                                     |

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data source: | Amazon Redshift, DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Teradata, Vertica, Yellowbrick                                                                                                                                                                                                                                                                                     |
| Notes:       | Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.<br>Support for Yellowbrick was added in SAS 9.4M7.                                                                                                                                                                                                                                                                                                                                                    |
| Tip:         | If you omit UPDATE_LOCK_TYPE=, you receive either the default action for the DBMS that you are using, or a lock for the DBMS that was specified with the LIBNAME statement. You can specify a lock for one DBMS table by using the data set option or for a group of DBMS tables by using the LIBNAME option.                                                                                                                                                                        |
| See:         | <a href="#">READ_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">READ_ISOLATION_LEVEL= data set option</a> , <a href="#">READ_LOCK_TYPE= LIBNAME option</a> , <a href="#">READ_LOCK_TYPE= data set option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= LIBNAME option</a> , <a href="#">UPDATE_ISOLATION_LEVEL= data set option</a> , <a href="#">UPDATE_LOCK_TYPE= LIBNAME option</a> , and DBMS-specific locking information in the reference section for your SAS/ACCESS interface |

## Syntax

**UPDATE\_LOCK\_TYPE=ROW | PAGE | TABLE | NOLOCK | VIEW**

### Syntax Description

#### **ROW**

locks a row if any of its columns are to be updated.

**Data source** Amazon Redshift, DB2 under UNIX and PC Hosts, Greenplum, HAWQ, Microsoft SQL Server, ODBC, OLE DB, Oracle, PostgreSQL, SAP HANA, Vertica

#### **PAGE**

locks a page of data. The number of bytes in a page is specific to the DBMS.

**Data source** SAP ASE

#### **TABLE**

locks the entire DBMS table.

**Data source** DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Microsoft SQL Server, ODBC, Oracle, SAP HANA, Teradata

#### **NOLOCK**

does not lock the DBMS table, page, or any rows when reading them for update.

**Data source** Amazon Redshift, Microsoft SQL Server, ODBC, Oracle, PostgreSQL, SAP ASE, SAP HANA

#### **VIEW**

locks the entire DBMS view.

**Data source** Teradata

---

## UPDATE\_MODE\_WAIT= Data Set Option

Specifies during SAS/ACCESS Update operations whether the DBMS waits to acquire a lock or fails your request when a different user has locked the DBMS resource.

|              |                                                                                     |
|--------------|-------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)            |
| Category:    | Data Access                                                                         |
| Default:     | LIBNAME option value                                                                |
| Data source: | Teradata                                                                            |
| See:         | <a href="#">UPDATE_MODE_WAIT= LIBNAME option. Locking in the Teradata Interface</a> |

---

## Syntax

**UPDATE\_MODE\_WAIT=YES | NO**

### Syntax Description

#### **YES**

specifies that Teradata waits to acquire the lock, so SAS/ACCESS waits indefinitely until it can acquire the lock.

#### **NO**

specifies that Teradata fails the lock request if the specified DBMS resource is locked.

---

## Details

If you specify UPDATE\_MODE\_WAIT=NO and if a different user holds a *restrictive* lock, then your SAS step fails and SAS/ACCESS continues the job by processing the next step. If you specify UPDATE\_MODE\_WAIT=YES, SAS/ACCESS waits indefinitely until it can acquire the lock.

A *restrictive* lock means that a different user is holding a lock that prevents you from obtaining your desired lock. Until the other user releases the restrictive lock, you cannot obtain your lock. For example, another user's table-level WRITE lock prevents you from obtaining a READ lock on the table.

Use SAS/ACCESS locking options only when Teradata standard locking is undesirable.

For more information, see the locking topic in the Teradata section.

---

## UPDATE\_SQL= Data Set Option

Determines which method to use to update and delete rows in a data source.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Default:     | LIBNAME option value                                                     |
| Data source: | Microsoft SQL Server, ODBC, PostgreSQL, Vertica                          |

See: [INSERT\\_SQL= data set option](#), [UPDATE\\_SQL= LIBNAME option](#)

---

## Syntax

**UPDATE\_SQL=YES | NO**

### Syntax Description

#### **YES**

specifies that SAS/ACCESS uses Current-of-Cursor SQL to update or delete rows in a table.

#### **NO**

specifies that SAS/ACCESS uses the SQLSetPos() API to update or delete rows in a table.

---

## Details

This is the update and delete equivalent of the [INSERT\\_SQL= data set option](#).

---

## UPDATEBUFF= Data Set Option

Specifies the number of rows that are processed in a single DBMS Update or Delete operation.

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software) |
| Category:    | Data Set Control                                                         |
| Default:     | LIBNAME option value                                                     |
| Data source: | Oracle                                                                   |

See: [UPDATEBUFF= LIBNAME option](#)

## Syntax

**UPDATEBUFF=***positive-integer*

### Syntax Description

***positive-integer***

is the maximum value that is allowed by the DBMS.

## Details

When updating with the VIEWTABLE window or PROC FSVIEW, use UPDATEBUFF=1 to prevent the DBMS interface from trying to update multiple rows. By default, these features use record-level locking and update only one observation at a time. They lock only the observation that is currently being edited.

---

## UPSERT= Data Set Option

Specifies whether a Teradata MultiLoad upsert should take place.

|              |                                                                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | PROC APPEND (when accessing DBMS data using SAS/ACCESS software)                                                                               |
| Category:    | Bulk Loading                                                                                                                                   |
| Default:     | NO                                                                                                                                             |
| Requirement: | MULTILOAD=YES must be specified.                                                                                                               |
| Data source: | Teradata                                                                                                                                       |
| See:         | <a href="#">MULTILOAD= data set option</a> , <a href="#">UPSERT_CONDITION= data set option</a> , <a href="#">UPSERT_WHERE= data set option</a> |

---

## Syntax

**UPSERT=**[YES](#) | NO

### Syntax Description

**YES**

uses the Teradata MultiLoad upsert feature.

**NO**

performs inserts without upserts.

## Details

The MultiLoad bulk-load facility supports the Teradata upsert feature. When an update fails because a target row does not exist, the upsert feature updates the table by inserting the missing row.

The *upsert* feature performs a combination of updates and inserts in one step. When updating a master table with a transaction table, an UPDATE statement is first issued on the master table using a row of data from the transaction table. If no target row exists to satisfy the update, an INSERT statement adds the transaction row to the master table.

Upsert requires a WHERE clause on the primary index of the target table to identify the target rows to be updated. When UPSERT=YES, Teradata builds a WHERE condition, by default, based on the primary index columns of the target table.

To add conditions to the WHERE condition, use [UPINSERT\\_CONDITION=](#).

---

## Example

```

libname x teradata user=user pw=pw;
/* Create a master table-Column where i is the primary index */
data x.master;
i=1;
j=1;
output;
i=2;
j=2;
output;
run;

/* Create a transaction table */
data x.transaction;
i=1;
j=99;
output;
run;

/* Running Upsert operates an Update statement based on the primary
index
column i. All other columns in the master are updated with transaction
data.
A snippet of the generated MultiLoad script is included:
.DML Label SASDML DO INSERT FOR MISSING UPDATE ROWS;
UPDATE "master"
SET "j"=:j";
WHERE "i"=:i";
INSERT "master"("i","j") VALUES (:i,:j");
*/
proc append base=x.master (MULTILOAD=YES UPSERT=YES)
data=x.transaction;
run;

```

---

## UPSERT\_CONDITION= Data Set Option

Specifies one or more conditions to add to a WHERE condition for a Teradata MultiLoad upsert.

Valid in: PROC APPEND only when appending Teradata tables (when using SAS/ACCESS software).

Categories: Bulk Loading  
Data Set Control

Default: none

Requirement: MULTILOAD=YES must be specified.

Data source: Teradata

See: [MULTILOAD= data set option](#), [UPSERT= data set option](#), [UPSERT\\_WHERE= data set option](#)

---

## Syntax

**UPSERT\_CONDITION=***condition(s)-to-append*

### Syntax Description

#### ***condition(s)-to-append***

specifies one or more conditions that are appended to [UPSERT\\_WHERE=](#) for upsert processing.

---

## Details

By default, MultiLoad upsert uses only the primary index columns of the target table to generate the WHERE condition for upsert processing.

When you need to add conditions to the generated WHERE condition, specify them using UPSERT\_CONDITION=. Add the AND keyword between the generated WHERE condition and the specified UPSERT\_CONDITION= data set option.

---

## Example

```
libname x teradata user=user pw=pw;
/* Create a master table-Column i where i is the primary index */
data x.master;
i=1;
j=1;
k=1;
```

```

output;
i=1;
j=2;
k=2;
output;
run;

/* Create a transaction table */
data x.transaction;
i=1;
j=99;output;
run;

/* Running Upsert with UPSERT_CONDITION= "j=2" causes "AND j=2" tp be
appended
to the Upsert Where clause. A snippet of the generated MultiLoad
script is included:

.DML Label SASDML DO INSERT FOR MISSING UPDATE ROWS;
UPDATE "master"
      SET "j"=:j", "k"="k";
WHERE "i"=:i" AND j=2;
INSERT "master"("i","j","k")  VALUES (:i",:j",:k");
*/
proc append base=x.master (MULTILOAD=YES UPSERT=YES
UPSERT_CONDITION="j=2")
      data=x.transaction;
run;

```

## UPSERT\_WHERE= Data Set Option

Specifies which columns in the master table are used to generate a WHERE condition for a Teradata MultiLoad upsert.

|              |                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | PROC APPEND (when accessing DBMS data using SAS/ACCESS software)                                                                         |
| Categories:  | Bulk Loading<br>Data Set Control                                                                                                         |
| Default:     | none                                                                                                                                     |
| Requirement: | MULTILOAD=YES must be specified.                                                                                                         |
| Data source: | Teradata                                                                                                                                 |
| See:         | <a href="#">MULTILOAD= data set option</a> , <a href="#">UPSERT= data set option</a> , <a href="#">UPSERT_CONDITION= data set option</a> |

## Syntax

**UPSERT\_WHERE=***list-of-column(s)*

## Syntax Description

### ***list-of-column(s)***

specifies one or more columns that are used to generate conditions for upsert processing of Teradata tables.

## Details

By default, MultiLoad upsert uses only the primary index columns of the target table to generate the WHERE condition for upsert processing.

Upsert processing requires a WHERE clause on the primary index of the target table to identify the target rows to be updated. When you are using the upsert feature, a WHERE condition is built, by default, based on the primary index columns of the target table.

When you need additional columns to identify target rows, use UPSERT\_WHERE= to list the columns to be used. Note that you need to include the columns that make up the primary index of the target table.

## Example

```

libname x teradata user=user pw=pw;
/* Create a master table-Column i where i is the primary index */
data x.master;
i=1;
j=1;
k=1;
output;
i=1;
j=2;
k=2;
output;
run;

/* Create a transaction table */
data x.transaction;
i=1;
j=2;
k=99;
output;
run;

/* Running Upsert with UPSERT_WHERE=(i j) generates an update Where
clause
based on columns i and j. All other columns in the master are updated
with
transaction data.

```

A snippet of the generated MultiLoad script is included:

```

.DML Label SASDML DO INSERT FOR MISSING UPDATE ROWS;
UPDATE "master"
```

```
      SET "k"=:k";
      WHERE "i"=:i" AND "j"=:j";
      INSERT "master"("i","j","k")  VALUES (:i,:j,:k);
      */

proc append base=x.master (MULTILOAD=YES    UPSERT=YES    UPSERT_WHERE=(i
j))
      data=x.transaction;
run;
```

---

## YB\_JAVA\_HOME= Data Set Option

Specifies the path to the JRE for Yellowbrick.

|              |                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | DATA and PROC steps (when accessing DBMS data using SAS/ACCESS software)                                                              |
| Category:    | Bulk Loading                                                                                                                          |
| Default:     | none                                                                                                                                  |
| Requirement: | This option is required for bulk loading to or bulk unloading from a Yellowbrick database.                                            |
| Data source: | Yellowbrick                                                                                                                           |
| See:         | <a href="#">BULKLOAD= data set option</a> , <a href="#">BULKUNLOAD= data set option</a> , <a href="#">BL_YB_PATH= data set option</a> |

---

## Syntax

**YB\_JAVA\_HOME=<'>*JRE-path*<'>**

### Required Argument

***JRE-path***

specifies the path to the Yellowbrick JRE. Enclose this value in single or double quotation marks if the path contains spaces or special characters.



# Macro Variables and System Options

---

|                                                                 |            |
|-----------------------------------------------------------------|------------|
| <i>Introduction to Macro Variables and System Options</i> ..... | <b>655</b> |
| <i>Macro Variables for Relational Databases</i> .....           | <b>656</b> |
| Automatic Macro Variables for SAS/ACCESS Processing .....       | 656        |
| Macro Variables for Passing Content to DBMS Queries .....       | 656        |
| <i>System Options for Relational Databases</i> .....            | <b>658</b> |
| Available System Options .....                                  | 658        |
| <i>Dictionary</i> .....                                         | <b>659</b> |
| DB2CATALOG= SAS System Option .....                             | 659        |
| DBFMTIGNORE SAS System Option .....                             | 660        |
| DBIDIRECTEXEC SAS System Option .....                           | 661        |
| DBSLICEPARM= SAS System Option .....                            | 664        |
| DBSRVTP= SAS System Option .....                                | 667        |
| SASTRACE= SAS System Option .....                               | 669        |
| SASTRACELOC= SAS System Option .....                            | 680        |
| SQLGENERATION= SAS System Option .....                          | 681        |
| SQLMAPPUTTO= SAS System Option .....                            | 685        |
| VALIDVARNAME= SAS System Option .....                           | 686        |

---

## Introduction to Macro Variables and System Options

This section describes [macro variables](#) on page 656 and [system options](#) on page 658 that you can use with SAS/ACCESS software. It describes only those components of the macro facility that depend on SAS/ACCESS engines. Most features of the SAS macro facility are portable.

For more information about the macro facility, see [SAS Macro Language: Reference](#). For more information about SAS system options, see [SAS System Options: Reference](#).

---

## Macro Variables for Relational Databases

---

### Automatic Macro Variables for SAS/ACCESS Processing

SYSDBMSG, SYSDBRC, SQLXMSG, and SQLXRC are automatic SAS macro variables. The SAS/ACCESS engine and your DBMS determine their values. Initially, SYSDBMSG and SQLXMSG are blank, and SYSDBRC and SQLXRC are set to 0.

SAS/ACCESS generates several return codes and error messages while it processes your programs. This information is available to you through these SAS macro variables.

#### SYSDBMSG

contains DBMS-specific error messages that are generated when you use SAS/ACCESS software to access your DBMS data.

#### SYSDBRC

contains DBMS-specific error codes that are generated when you use SAS/ACCESS software to access your DBMS data. Error codes that are returned are text, not numbers.

You can use these variables anywhere while you are accessing DBMS data. Only one set of macro variables is provided, however. So it is possible that, if tables from two different DBMSs are accessed, it might not be clear from which DBMS the error message originated. To address this problem, the name of the DBMS is inserted at the beginning of the SYSDBMSG macro variable message or value. The contents of the SYSDBMSG and SYSDBRC macro variables can be printed in the SAS log by using the %PUT macro. They are reset after each SAS/ACCESS LIBNAME statement, DATA step, or procedure is executed. In the statement below, %SUPERQ masks special characters such as &, %, and any unbalanced parentheses or quotation marks that might exist in the text stored in the SYSDBMSG macro.

```
%put %superq(SYSDBMSG)
```

These special characters can cause unpredictable results if you use this statement:

```
%put &SYSDBMSG
```

It is more advantageous to use %SUPERQ.

If you try to connect to Oracle and use the incorrect password, you receive the messages shown in this output.

**Output 14.1 SAS Log for an Oracle Error**

```

2? libname mydblib oracle user=pierre pass=paris path="mypath";

ERROR: Oracle error trying to establish connection. Oracle error is
      ORA-01017: invalid username/password; logon denied
ERROR: Error in the LIBNAME or FILENAME statement.
3? %put %superq(sysdbmsg);

Oracle: ORA-01017: invalid username/password; logon denied
4? %put &sysdbrc;

-1017
5?

```

You can also use SYMGET to retrieve error messages:

```
msg=symget ("SYSDBMSG");
```

Here is an example.

```

data_null_;
msg=symget ("SYSDBMSG");
put msg;
run;

```

The SQL pass-through facility generates return codes and error messages that are available to you through these SAS macro variables:

**SQLXMSG**  
contains DBMS-specific error messages.

**SQLXRC**  
contains DBMS-specific error codes.

You can use SQLXMSG and SQLXRC through implicit or explicit pass-through with the SQL pass-through facility. See [Return Codes on page 691](#).

You can print the contents of SQLXMSG and SQLXRC in the SAS log by using the %PUT macro. SQLXMSG is reset to a blank string, and SQLXRC is reset to 0 when any SQL pass-through facility statement is executed.

## Macro Variables for Passing Content to DBMS Queries

With some data sources, you can pass extra content to the data source when you pass an explicit SQL query with PROC SQL. To do this, you enable the **PRESERVE\_COMMENTS= LIBNAME** option and pass the content in a comment. The comment can contain keywords or markup that is used by the query processor for your data source.

For the same data sources that use the PRESERVE\_COMMENTS= LIBNAME option, you can pass similar content at the beginning of SQL statements that are generated by SAS for other operations, such as a call to PROC PRINT. To do this, you define a macro variable that corresponds to your data source. The text that you assign to the macro variable is added at the front of each generated SQL command that is passed to the data source. This text can contain keywords or markup that is

used by the query processor for your data source. To see the query that is passed to the data source, including any text that you have added, specify `SASTRACE=' , , ,d'` in the OPTIONS statement.

**Table 14.1** Macro Variables that Pass Content to Generated SQL Statements

| Data Source <sup>1</sup> | Macro Variable        |
|--------------------------|-----------------------|
| DB2                      | DB2_SQL_COMMENT       |
| Microsoft SQL Server     | SQLSERVER_SQL_COMMENT |
| ODBC                     | ODBC_SQL_COMMENT      |

<sup>1</sup> There is also a macro variable that is used for SAS/ACCESS to PC Files. For more information, see [SAS/ACCESS Interface to PC Files: Reference](#).

For example, if you want to add content to the front of a query before a call to PROC PRINT, you might enter the following code. In this case, the Db2lib libref connects to a DB2 data source.

```
%let db2_sql_comment=<OPTGUIDELINES> <REGISTRY> <OPTION NAME='DB2_UNION_OPTIMIZATION' VALUE='DISABLE_OJPPD_FOR_NP' /> </REGISTRY> </OPTGUIDELINES>; ;
PROC PRINT data=db2lib.cars; run;
```

The text string that you assign to the db2\_sql\_comment variable is added to the beginning of the SELECT statement that SAS generates from the call to PROC PRINT. The resulting query that is passed to the DB2 data source would be:

```
/* <OPTGUIDELINES> <REGISTRY> <OPTION NAME='DB2_UNION_OPTIMIZATION' VALUE='DISABLE_OJPPD_FOR_NP' /> </REGISTRY> </OPTGUIDELINES>; */ SELECT * from CARS
```

## System Options for Relational Databases

### Available System Options

Here are the available systems options.

No SAS/ACCESS interface support is available for the REPLACE= system option. (See [SAS System Options: Reference](#).)

| SAS System Options            | Default         |
|-------------------------------|-----------------|
| <a href="#">DB2CATALOG=</a>   | SYSIBM          |
| <a href="#">DBFMTIGNORE</a>   | NODBFMTIGNORE   |
| <a href="#">DBIDIRECTEXEC</a> | NODBIDIRECTEXEC |

| SAS System Options | Default                                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DBSLICEPARM=       | THREADED_APPS,2 [DB2 under z/OS, Oracle, and Teradata]<br>THREADED_APPS,2 or THREADED_APP,3 [DB2 under UNIX and PC Hosts, Informix, Microsoft SQL Server, ODBC, and SAP ASE, SAP IQ] |
| DBSRVTP=           | NONE                                                                                                                                                                                 |
| REPLACE=           | No SAS/ACCESS interface support is available. See <a href="#">SAS System Options: Reference</a> .                                                                                    |
| SASTRACE=          | none                                                                                                                                                                                 |
| SASTRACELOC=       | stdout                                                                                                                                                                               |
| SQLGENERATION=     | Specifies whether and when SAS procedures generate SQL for in-database processing of source data.                                                                                    |
| SQLMAPPUTTO=       | Specifies whether the PUT function is mapped to the SAS_PUT() function for a database, possibly also where the SAS_PUT() function is mapped.                                         |
| VALIDVARNAME=      | Controls the type of SAS variable names that can be used or created during a SAS session.                                                                                            |

## Dictionary

### DB2CATALOG= SAS System Option

Overrides the default owner of DB2 catalog tables.

- Valid in:            OPTIONS statement  
 Category:          Databases: DB2  
 Default:            SYSIBM

|              |                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Restriction: | This option applies to only the local DB2 subsystem. So when you specify the LOCATION= or SERVER= connection option in the LIBNAME statement, the SAS/ACCESS engine always uses SYSIBM as the default value. |
| Data source: | DB2 under z/OS                                                                                                                                                                                               |
| See:         | <a href="#">LOCATION= connection option</a> , <a href="#">SERVER= connection option</a>                                                                                                                      |

## Syntax

**DB2CATALOG=SYSIBM | catalog-owner**

### Required Arguments

#### SYSIBM

specifies the default catalog owner.

#### catalog-owner

specifies a different catalog owner from the default.

## Details

The default value for this option is initialized when SAS is installed. You can override the default only when these conditions are met:

- SYSIBM cannot be the owner of the catalog that you want to access.
- Your site must have a shadow catalog of tables (one to which all users have access).
- You must specify DB2CATALOG= in the restricted options table and then rebuild the table.

## DBFMTIGNORE SAS System Option

Specifies whether to ignore numeric formats.

|              |                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------|
| Valid in:    | configuration file, SAS invocation, OPTIONS statement,SAS System Options window                                 |
| Categories:  | Files: SAS Files<br>Input Control: Data Processing                                                              |
| Default:     | NODBFMTIGNORE                                                                                                   |
| Restriction: | This option pertains only to SAS formats that are numeric.                                                      |
| Data source: | Teradata                                                                                                        |
| See:         | <a href="#">SQL_FUNCTIONS= LIBNAME option</a> , <a href="#">SAS In-Database Products: Administrator's Guide</a> |

# Syntax

## **DBFMTIGNORE | NODBFMTIGNORE**

### Required Arguments

#### **DBFMTIGNORE**

specifies that numeric formats are ignored and FLOAT data type is created.

#### **NODBFMTIGNORE**

specifies that numeric formats are used.

---

## Details

You normally use numeric formats to specify a database data type when processing output. SAS takes all nonnumeric formats (such as DATE, TIME, DATETIME, and CHAR) as hints when it processes output. So use this option to ignore numeric formats and create a FLOAT data type instead. For example, the SAS/ACCESS engine creates a table with a column type of INT for a SAS variable with a format of 5.0.

---

## DBIDIRECTEXEC SAS System Option

Lets the SQL pass-through facility optimize handling of SQL statements by passing them directly to the database for execution.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Categories: Files: External Files

System Administration: Performance

Default: DBIDIRECTEXEC

Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica, Yellowbrick

Notes: Support for Amazon Redshift was added in the April 2016 release of SAS/ACCESS.

Support for JDBC was added in SAS Viya 3.4.

Support for Microsoft SQL Server was added in SAS 9.4M6.

Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS.

Support for Spark was added in SAS 9.4M7.

Support for Yellowbrick was added in SAS 9.4M7.

Support for ODBC was added in SAS 9.4M8.

Tip: Use the SQLPONEATTEMPT system option, SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT environment variable, or both to further

control how often you attempt to pass SQL statements to the database. See “[Control Attempts to Pass SQL to the Database](#)”.

See: [SQLPONEATTEMPT](#) system option

## Syntax

**DBIDIRECTEXEC | NODBIDIRECTEXEC**

### Required Arguments

#### **DBIDIRECTEXEC**

indicates that the SQL pass-through facility optimizes handling of SQL statements by passing them directly to the database for execution, which optimizes performance. Using this option, you can process CREATE TABLE AS SELECT, INSERT INTO TABLE, INSERT INTO VALUES, UPDATE, and DELETE statements.

#### **NODBIDIRECTEXEC**

indicates that the SQL pass-through facility does not optimize handling of SQL statements.

## Details

### About DBIDIRECTEXEC

You can specify DBIDIRECTEXEC to significantly improve CPU, input, and output performance. The DBIDIRECTEXEC system option applies to specific SQL statements that you specify when using PROC SQL.

Certain database-specific criteria must be met to pass SQL statements to the DBMS. These criteria are the same as the criteria that exist for passing joins. For details for your DBMS, see “[Passing Joins to the DBMS](#) on page 61” and “[When Passing Joins to the DBMS Fails](#)” on page 64.

When the required criteria for passing SQL to the database are met, a database can process the CREATE TABLE *table-name* AS SELECT statement in a single step instead of as three separate statements (CREATE, SELECT, and INSERT). That is, when you enable DBIDIRECTEXEC, PROC SQL sends the CREATE TABLE AS SELECT statement to the database.

Note that when DBIDIRECTEXEC is enabled, any LENGTH= column modifier that is specified in the CREATE TABLE *table-name* AS SELECT statement is ignored. To use the LENGTH= column modifier, you must set NODBIDIRECTEXEC.

You can also send a DELETE, INSERT INTO TABLE, INSERT INTO VALUES, or UPDATE statement directly to the database for execution, which can improve CPU, input, and output performance.

Before an SQL statement can be processed, all librefs that are associated with the statement must reference compatible data sources. For example, a CREATE TABLE AS SELECT statement that creates a table by selecting from a SAS table is

not sent to the database for execution. The reason is that the data sources are not compatible. The libref must also use the same database server for all compatible data sources.

If multiple librefs point to different data sources, the statement is processed as if NODBIDIRECTEXEC were specified, regardless of how you specified this option.

Once a system administrator specifies the default for this option globally, users can override it within their own configuration file.

## Control Attempts to Pass SQL to the Database

Use the SQLPONEATTEMPT system option force PROC SQL to stop additional attempts to pass part of an SQL query to the database if the first attempt to pass a query fails. If NOSQLPONEATTEMPT is specified, then PROC SQL attempts to break a query into smaller parts that are then passed to the database. PROC SQL can attempt to pass SQL statements to the database until the statements are accepted by the database or until the query cannot be broken down further. Data might be passed between the database and SAS for each attempt to break a query into smaller parts.

The SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT environment variable prevents multiple attempts to pass an SQL query to the database during run time. When you set SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT=YES, you improve performance by preventing the transfer of potentially large amounts of data into SAS for further processing. The SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT environment variable is applicable to the CREATE TABLE AS ... SELECT, INSERT INTO TABLE, INSERT INTO VALUES, UPDATE, and DELETE statements.

The SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT environment variable complements the SQLPONEATTEMPT system option. SQLPONEATTEMPT stops processing a query if it detects at compile time that the full query would not pass successfully to the database. SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT stops processing if a query fails to pass through to the database during run time. By preventing transfer of data into SAS, you can then modify your query so that it can pass successfully to the database for faster processing.

To set the SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT environment variable, specify SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT=YES as required for your operating environment. Alternatively, you can use the SET system option to set SQL\_DBIDIRECTEXEC\_IP\_ONE\_ATTEMPT in a SAS session, as shown here:

```
options set= SQL_DBIDIRECTEXEC_IP_ONE_ATTEMPT=YES;
```

## Examples

### Example 1: Create a Temporary Table

This example creates a temporary table from a SELECT statement using the DBIDIRECTEXEC system option.

```
libname lib1 db2 user=myusr1 password=mypwd1 datasrc=sample
connection=global;
libname lib2 db2 user=myusr2 password=mypwd2 datasrc=sample
connection=global dbmstemp=yes;
```

```

data lib1.tab1;
  a=1;
  b='one';
run;
options dbidirectexec sastraceloc=saslog;
proc sql;
  create table lib2.tab1 as
    select * from lib1.tab1;
quit;

```

## Example 2: Reference One Database, Use Different Schemas

In this example, two librefs point to the same database server but use different schemas.

```

libname lib1 db2 user=myusr1 password=mypwd1 datasrc=sample;
libname lib2 db2 user=myusr2 password=mypwd2 datasrc=sample;
data lib1.tab1;
  a=1;
  b='one';
run;
options dbidirectexec sastraceloc=saslog;
proc sql;
  create table lib2.tab2 as
    select * from lib1.t1;
quit;

```

## Example 3: Pass a Statement Directly to the Database

This example shows how a statement can be passed directly to the database for execution, if you specify DBIDIRECTEXEC.

```

libname company oracle user=myusr1 pw=mypwd1 path=mydb;
proc sql;
  create table company.hr_tab as
    select * from company.emp
    where deptid = 'HR';
quit;

```

## DBSLICEPARM= SAS System Option

Controls the scope of DBMS threaded Reads and the number of threads.

|           |                                                                                             |
|-----------|---------------------------------------------------------------------------------------------|
| Valid in: | configuration file, SAS invocation, OPTIONS statement, SAS System Options window            |
| Category: | System Administration: Performance                                                          |
| Defaults: | THREADED_APPS, none [Greenplum, HAWQ]<br>THREADED_APPS,2 [DB2 under z/OS, Oracle, Teradata] |

THREADED\_APPS,2 or THREADED\_APPS,3 [DB2 under UNIX and PC Hosts, Informix, Microsoft SQL Server, ODBC, SAP ASE, SAP IQ]

Data source: DB2 under UNIX and PC Hosts, DB2 under z/OS, Greenplum, HAWQ, Informix, Microsoft SQL Server, ODBC, Oracle, SAP ASE, SAP IQ, Teradata

Notes: Support for Greenplum was added in SAS 9.4M2.  
Support for HAWQ was added in SAS 9.4M3.

See: [DBSLICE= data set option](#), [DBSLICEPARM= LIBNAME option](#), [DBSLICEPARM= data set option](#), [SLEEP= LIBNAME option](#), [SLEEP= data set option](#), [TENACITY= LIBNAME option](#), [TENACITY= data set option](#)

## Syntax

```
DBSLICEPARM=NONE | THREADED_APPS | ALL
DBSLICEPARM=(NONE | THREADED_APPS | ALL <max-threads> )
DBSLICEPARM=(NONE | THREADED_APPS | ALL<, max-threads>)
```

## Required Arguments

### **NONE**

disables DBMS threaded Read. SAS reads tables on a single DBMS connection, as it did with SAS 8 and earlier.

### **THREADED\_APPS**

makes fully threaded SAS procedures (threaded applications) eligible for threaded Reads.

### **ALL**

makes all read-only librefs eligible for threaded Reads. This includes SAS threaded applications, as well as the SAS DATA step and numerous SAS procedures.

### ***max-threads***

a positive integer value that specifies the maximum number of connections per table read. The second parameter of the option determines the number of threads to read the table in parallel. The number of partitions on the table determine the number of connections made to the Oracle server for retrieving rows from the table. A partition or portion of the data is read on each connection. The combined rows across all partitions are the same regardless of the number of connections. That is, changes to the number of connections do not change the result set. Increasing the number of connections instead redistributes the same result set across more connections.

There are diminishing returns when increasing the number of connections. With each additional connection, more burden is placed on the DBMS, and a smaller percentage of time saved on the SAS step. See the DBMS-specific reference section for details about partitioned reads before using this parameter.

## Details

You can use DBSLICEPARM= in numerous locations. The usual rules of option precedence apply: A table option has the highest precedence, then a LIBNAME option, and so on. SAS configuration file option has the lowest precedence because DBSLICEPARM= in any of the other locations overrides that configuration value.

DBSLICEPARM=ALL and DBSLICEPARM=THREADED\_APPS make SAS programs eligible for threaded Reads. To see whether threaded Reads are actually generated, turn on SAS tracing and run a program, as shown in this example.

```
options sastrace=",t" sastraceloc=saslog nostsuffix;
proc print data=lib.dbtable(dbsliceparm=(ALL));
  where dbcol>1000;
run;
```

If you want to directly control the threading behavior, use the DBSLICE= data set option.

*Greenplum, HAWQ:* There is no default value for the maximum number of connections per table read. This value depends on the number of partitions in a table and on arguments that are used with the MOD function in a WHERE clause. For more information, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page [76](#).

*DB2 under UNIX and PC Hosts, Informix, Microsoft SQL Server, ODBC, SAP ASE, SAP IQ:* The default thread number depends on whether an application passes in the number of threads (CPUCOUNT=) and whether the data type of the column that was selected for data partitioning is binary.

---

## Examples

### Example 1: Disable Threaded Read for All SAS Users

Here is how to use DBSLICEPARM= in a SAS configuration file entry in Windows to turn off threaded Reads for all SAS users.

```
-dbsliceparm NONE
```

### Example 2: Enable Threaded Reads for Read-Only References

Here is how you can use DBSLICEPARM= as a z/OS invocation option to turn on threaded Reads for read-only references to DBMS tables throughout a SAS job.

```
sas o(dbsliceparm=ALL)
```

## Example 3: Increase Maximum Threads (as a SAS Global Option)

```
option dbsliceparm=(threaded_apps,3);
```

## Example 4: Enable Threaded Reads for References Using a Particular Libref

You can use DBSLICEPARM= as a LIBNAME option to turn on threaded Reads for read-only table references that use this particular libref, as shown in this example.

```
libname dblib oracle user=myusr1 password=mypwd1 dbsliceparm=ALL;
```

## Example 5: Enable Threaded Reads as a Table-Level Option

Here is how to use DBSLICEPARM= as a table-level option to turn on threaded Reads for this particular table, requesting up to four connections.

```
proc reg SIMPLE;
  data=dbllib.customers (dbsliceparm=(all,4));
  var age weight;
  where years_active>1;
run;
```

## DBSRVTP= SAS System Option

Specifies whether SAS/ACCESS engines hold or block the originating client while making performance-critical calls to the database.

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| Valid in:    | SAS invocation                                                                          |
| Category:    | Communications: Networking and Encryption                                               |
| Default:     | NONE                                                                                    |
| Data source: | DB2 under UNIX and PC Hosts, Informix, Netezza, ODBC, OLE DB, Oracle, SAP ASE, Teradata |

## Syntax

**DBSRVTP='ALL' | 'NONE' | '(engine-name(s))'**

## Required Arguments

### ALL

indicates that SAS does not use any blocking operations for all underlying SAS/ACCESS engines that support this option.

**NONE**

indicates that SAS uses standard blocking operations for all SAS/ACCESS engines.

***engine-name(s)***

indicates that SAS does not use any blocking operations for the specified SAS/ACCESS engines. You can specify one or more engine names. If you specify more than one engine name, separate them with blank spaces and enclose the list in parentheses.

db2 (under UNIX and PC Hosts only)

informix

netezza

odbc (indicates that SAS uses non-blocking operations for SAS/ACCESS ODBC and Microsoft SQL Server interfaces)

oledb

oracle

sapase

teradata (not supported on z/OS)

## Details

This option applies only when SAS is called as a server responding to multiple clients.

You can use this option to help throughput of the SAS server because it supports multiple simultaneous execution streams, if the server uses certain SAS/ACCESS interfaces. Improved throughput occurs when the underlying SAS/ACCESS engine does not hold or block the originating client. That is, any one client using a SAS/ACCESS product does not keep the SAS server from responding to other client requests. SASSHARE software and SAS Integration Technologies are two ways of invoking SAS as a server.

This option is a system invocation option, which means the value is specified when SAS is invoked. Because the DBSRVTP= option uses multiple native threads, enabling this option uses the underlying DBMS's threading support. Some databases handle threading better than others, so you might want to invoke DBSRVTP= for some DBMSs and not others. Refer to your documentation for your DBMS for more information.

The option accepts a string where values are the engine name of a SAS/ACCESS product, ALL, or NONE. When specifying multiple values, enclose the values in quotation marks and parentheses, and separate the values with a space.

This option is applicable on all Windows platforms, AIX, Solaris, and z/OS (Oracle only). On some of these hosts, you can call SAS with the -SETJMP system option. Specifying -SETJMP disables the DBSRVTP= option.

## Example

Each of these examples calls SAS from the UNIX command line.

```
sas -dbsrvtp all
sas -dbsrvtp '(oracle db2)'
sas -dbsrvtp teradata
sas -dbsrvtp '(sapase informix odbc oledb)'
sas -dbsrvtp none
```

## SASTRACE= SAS System Option

Generates trace information from a DBMS engine.

|              |                                                                                                                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | configuration file, SAS invocation, OPTIONS statement                                                                                                                                                                                                              |
| Category:    | Log and Procedure Output Control: SAS Log                                                                                                                                                                                                                          |
| Default:     | none                                                                                                                                                                                                                                                               |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica |
| Notes:       | Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS<br>Support for Spark was added in SAS 9.4M7.                                                                                                                          |
| Tip:         | You can also use more than one SASTRACE= option at a time if the arguments are for different positions in the argument list (for example, SASTRACE=',,d,d'). The value 'd,' cannot be combined with other values.                                                  |
| See:         | <a href="#">SASTRACELOC= system option</a> , Generating Trace Information for Threaded Reads on page 72                                                                                                                                                            |

## Syntax

**SASTRACE=',,d' | ',,d,' | 'd,' | 'd,,,' | ',,db' | ',,s' | ',,sa' | ',,t,' | OFF**

## Required Arguments

**',,d'**

specifies that all SQL statements that are sent to the DBMS are sent to the log. Here are the applicable statements:

|        |                |
|--------|----------------|
| SELECT | DELETE         |
| CREATE | SYSTEM CATALOG |
| DROP   | COMMIT         |
| INSERT | ROLLBACK       |
| UPDATE |                |

For engines that do not generate SQL statements, API calls and all parameters are sent to the log.

',,d,'

specifies that all routine calls are sent to the log. All function enters, exits, and pertinent parameters and return codes are traced when you select this option. The information varies from engine to engine, however.

This option is most useful if you have a problem and need to send a SAS log to technical support for troubleshooting.

'd,'

specifies that all DBMS calls (such as API and client calls, connection information, column bindings, column error information, and row processing) are sent to the log. This information varies from engine to engine, however.

This option is most useful if you have a problem and need to send a SAS log to technical support for troubleshooting.

'd,''

specifies that version information for the current DBMS and client are displayed in the log.

',,db'

specifies that only a brief version of all SQL statements that the ',,d' option normally generates are sent to the log.

',,s'

specifies that a summary of timing information for calls made to the DBMS is sent to the log.

',,sa'

specifies that timing information for each call that is made to the DBMS is sent to the log along with a summary.

',,t,'

specifies that all threading information is sent to the log. Here is the information that it includes:

- number of threads that are spawned
- number of observations that each thread contains
- exit code of the thread, if it fails

**OFF**

specifies to turn tracing off.

## Details

**SASTRACE=** and **SASTRACELOC=** behavior is specific to SAS/ACCESS software. **SASTRACE=** is a very powerful tool to use when you want to see the commands that SAS/ACCESS sent to your DBMS. **SASTRACE=** output is DBMS-specific. However, most SAS/ACCESS engines show you statements like **SELECT** or **COMMIT** as the DBMS processes them for the SAS application. These details can help you manage **SASTRACE=** output in your DBMS.

- When using **SASTRACE=** on PC platforms, you must also specify **SASTRACELOC=**.
- Here is how to turn SAS tracing off:

```
options sastrace=off;
```

- Log output is much easier to read if you specify NOSTSUFFIX. Because this code is entered without specifying the option, the resulting log is longer and harder to decipher.

---

**Note:** NOSTSUFFIX is not supported on z/OS.

---

```
options sastrace=',,,d' sastraceloc=saslog;
proc print data=mydblib.snow_birthdays;
run;
```

Here is the resulting log.

```
0 1349792597 sastb_next 2930 PRINT
ORACLE_5: Prepared: 1 1349792597 sastb_next 2930 PRINT
SELECT * FROM scott.SNOW_BIRTHDAYS 2 1349792597 sastb_next 2930 PRINT
3 1349792597 sastb_next 2930 PRINT
16 proc print data=mydblib.snow_birthdays; run;
4 1349792597 sastb_next 2930 PRINT
ORACLE_6: Executed: 5 1349792597 sastb_next 2930 PRINT
Prepared statement ORACLE_5 6 1349792597 sastb_next 2930 PRINT
7 1349792597 sastb_next 2930 PRINT
```

Use NOSTSUFFIX to make the log file much easier to read.

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix;
proc print data=mydblib.snow_birthdays;
run;
```

Here is the resulting log.

```
ORACLE_1: Prepared:
SELECT * FROM scott.SNOW_BIRTHDAYS
12 proc print data=mydblib.snow_birthdays; run;
ORACLE_2: Executed:
Prepared statement ORACLE_1
```

---

## Examples

### Example 1: Use SQL Trace ',,,d'

These examples use NOSTSUFFIX and SASTRACELOC=SASLOG.

```
data work.winter_birthdays;
  input empid birthdat date9. lastname $18.;
  format birthdat date9..;
datalines;
678999 28DEC1966 PAVEO      JULIANA    3451
456788 12JAN1977 SHIPTON    TIFFANY    3468
890123 20FEB1973 THORSTAD   EDWARD    3329
;
run;
```

Examples are based on this data set.

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix;
libname mydblib oracle user=myusr1 password=mypwd1 schema=bday_data;
```

```
data mydblib.snow_birthdays;
   set work.winter_birthdays;
run;
libname mydblib clear;
```

Output for this ',,d' example is written to the SAS log, as specified in the  
SASTRACELOC=SASLOG option.

**Output 14.2 SAS Log Output from the SASTRACE=',,d' System Option**

```

30   data work.winter_birthdays;
31     input empid birthdat date9. lastname $18.;
32     format birthdat date9.1;
33   datalines;
NOTE: The data set WORK.WINTER_BIRTHDAYS has 3 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time          0.03 seconds
      cpu time          0.04 seconds
37   ;
38   run;
39   options sastrace=',,d' sastraceloc=saslog nostsuffix;
40   libname mydblib oracle user=myusr1 password=XXXXX schema=bdy_data;
NOTE: Libref MYDBLIB was successfully assigned as follows:
      Engine:          ORACLE
      Physical Name:
41   proc datasets library=mydblib;
      delete snow_birthdays;run;
ORACLE_1: Prepared:
SELECT * FROM SNOW_BIRTHDAYS
ORACLE_2: Executed:
DROP TABLE SNOW_BIRTHDAYS
NOTE: Deleting MYDBLIB.SNOW_BIRTHDAYS (memtype=DATA) .
NOTE: PROCEDURE DELETE used (Total process time):
      real time          0.26 seconds
      cpu time          0.12 seconds
42   data mydblib.snow_birthdays;
43     set work.winter_birthdays;
44   run;
ORACLE_3: Prepared:
SELECT * FROM SNOW_BIRTHDAYS
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
ORACLE_4: Executed:
CREATE TABLE SNOW_BIRTHDAYS(empid NUMBER ,birthdat DATE,lastname VARCHAR2 (18))
ORACLE_5: Prepared:
INSERT INTO SNOW_BIRTHDAYS (empid,birthdat,lastname) VALUES
(:empid,TO_DATE(:birthdat,'DDMONYYYY','NLS_DATE_LANGUAGE=American'),:lastname)
NOTE: There were 3 observations read from the data set WORK.WINTER_BIRTHDAYS.
ORACLE_6: Executed:
Prepared statement ORACLE_5
ORACLE: *-*-*-*-*-* COMMIT *-*-*-*-*-*-
NOTE: The data set MYDBLIB.SNOW_BIRTHDAYS has 3 observations and 3 variables.
ORACLE: *-*-*-*-*-* COMMIT *-*-*-*-*-
NOTE: DATA statement used (Total process time):
      real time          0.47 seconds
      cpu time          0.13 seconds
ORACLE_7: Prepared:
SELECT * FROM SNOW_BIRTHDAYS
45   proc print data=mydblib.snow_birthdays; run;
ORACLE_8: Executed:
Prepared statement ORACLE_7
NOTE: There were 3 observations read from the data set MYDBLIB.SNOW_BIRTHDAYS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.04 seconds
      cpu time          0.04 seconds
46
47   libname mydblib clear;
NOTE: Libref MYDBLIB has been deassigned.

```

**Example 2: Use Log Trace ',,d'**

```

options sastrace=',,d,' sastraceloc=saslog nostsuffix;
libname mydblib oracle user=myusr1 password=mypwd1 schema=bdy_data;

```

```
data mydblib.snow_birthdays;
   set work.winter_birthdays;
run;
libname mydblib clear;
```

Output is written to the SAS log, as specified in the SASTRACELOC=SASLOG option.

**Output 14.3 SAS Log Output from the SASTRACE=',,d,' System Option**

```

84   options sastrace=',,d,' sastraceloc=saslog nostsuffix;
ACCESS ENGINE: Entering DBICON
ACCESS ENGINE: Number of connections is 1
ORACLE: orcon()
ACCESS ENGINE: Successful physical conn id 1
ACCESS ENGINE: Exiting DBICON, Physical Connect id = 1, with rc=0X00000000
85   libname mydblib oracle user=myusr1 password=mypwd1 schema=bday_data;
ACCESS ENGINE: CONNECTION= SHAREDREAD
NOTE: Libref MYDBLIB was successfully assigned as follows:
      Engine:          ORACLE
      Physical Name:  lupin
86   data mydblib.snow_birthdays;
87       set work.winter_birthdays;
88   run;
ACCESS ENGINE: Entering yoeopen
ACCESS ENGINE: Entering dbiopen
ORACLE: oropen()
ACCESS ENGINE: Successful dbiopen, open id 0, connect id 1
ACCESS ENGINE: Exit dbiopen with rc=0X00000000
ORACLE: orgall()
ORACLE: orprep()
ACCESS ENGINE: Entering dbiclose
ORACLE: orcclose()
ACCESS ENGINE: DBICLOSE open_id 0, connect_id 1
ACCESS ENGINE: Exiting dbiclos with rc=0X00000000
ACCESS ENGINE: Access Mode is XO_OUTPUT
ACCESS ENGINE: Access Mode is XO_SEQ
ACCESS ENGINE: Shr flag is XHSHRMEM
ACCESS ENGINE: Entering DBICON
ACCESS ENGINE: CONNECTION= SHAREDREAD
ACCESS ENGINE: Number of connections is 2
ORACLE: orcon()
ACCESS ENGINE: Successful physical conn id 2
ACCESS ENGINE: Exiting DBICON, Physical Connect id = 2, with rc=0X00000000
ACCESS ENGINE: Entering dbiopen
ORACLE: oropen()
ACCESS ENGINE: Successful dbiopen, open id 0, connect id 2
ACCESS ENGINE: Exit dbiopen with rc=0X00000000
ACCESS ENGINE: Exit yoeopen with SUCCESS.
ACCESS ENGINE: Begin yoeinfo
ACCESS ENGINE: Exit yoeinfo with SUCCESS.
ORACLE: orovar()
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
ORACLE: oroload()
ACCESS ENGINE: Entering dbrload with SQL Statement set to
              CREATE TABLE SNOW_BIRTHDAYS(empid NUMBER ,birthdat DATE,lastname VARCHAR2
(18))
ORACLE: orexec()
ORACLE: orexec() END
ORACLE: orins()
ORACLE: orubuf()
ORACLE: orubuf()
ORACLE: SAS date : 28DEC1966
ORACLE: orins()
ORACLE: SAS date : 12JAN1977
ORACLE: orins()
ORACLE: SAS date : 20FEB1973
NOTE: There were 3 observations read from the data set WORK.WINTER_BIRTHDAYS.

```

```

ORACLE: orforc()
ORACLE: orflush()
NOTE: The data set MYDBLIB.SNOW_BIRTHDAYS has 3 observations and 3 variables.
ACCESS ENGINE: Enter yoeclos
ACCESS ENGINE: Entering dbiclose
ORACLE: orcclose()
ORACLE: orforc()
ORACLE: orflush()
ACCESS ENGINE: DBICLOSE open_id 0, connect_id 2
ACCESS ENGINE: Exiting dbiclos with rc=0X00000000
ACCESS ENGINE: Entering DBIDCON
ORACLE: ordcon
ACCESS ENGINE: Physical disconnect on id = 2
ACCESS ENGINE: Exiting DBIDCON with rc=0X00000000, rc2=0X00000000
ACCESS ENGINE: Exit yoeclos with rc=0x00000000
NOTE: DATA statement used (Total process time):
      real time          0.21 seconds
      cpu time           0.06 seconds
ACCESS ENGINE: Entering DBIDCON
ORACLE: ordcon
ACCESS ENGINE: Physical disconnect on id = 1
ACCESS ENGINE: Exiting DBIDCON with rc=0X00000000, rc2=0X00000000
89   libname mydblib clear;
NOTE: Libref MYDBLIB has been deassigned.

```

### Example 3: Use DBMS Trace 'd,'

```

options sastrace='d,' sastraceloc=saslog nostsuffix;
libname mydblib oracle user=myusr1 password=mypwd1 schema=bday_data;
data mydblib.snow_birthdays;
  set work.winter_birthdays;
run;
libname mydblib clear;

```

Output is written to the SAS log, as specified in the SASTRACELOC=SASLOG option.

**Output 14.4 SAS Log Output from the SASTRACE= 'd,' System Option**

```

ORACLE: PHYSICAL connect successful.
ORACLE: USER=myusr1
ORACLE: PATH=mypath
ORACLE: SCHEMA=bday_data
110 libname mydblib oracle user=myusr1 password=mypwd1 path=mypath
schema=bday_data;
NOTE: Libref MYDBLIB was successfully assigned as follows:
      Engine:          ORACLE
      Physical Name:  mypath
111  data mydblib.snow_birthdays;
112    set work.winter_birthdays;
113  run;
ORACLE: PHYSICAL connect successful.
ORACLE: USER=myusr1
ORACLE: PATH=mypath
ORACLE: SCHEMA=bday_data
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
ORACLE: INSERTBUFF option value set to 10.
NOTE: There were 3 observations read from the data set WORK.WINTER_BIRTHDAYS.
ORACLE: Rows processed: 3
ORACLE: Rows failed : 0
NOTE: The data set MYDBLIB.SNOW_BIRTHDAYS has 3 observations and 3 variables.
ORACLE: Successfully disconnected.
ORACLE: USER=myusr1
ORACLE: PATH=mypath
NOTE: DATA statement used (Total process time):
      real time        0.21 seconds
      cpu time         0.04 seconds
ORACLE: Successfully disconnected.
ORACLE: USER=myusr1
ORACLE: PATH=mypath
114 libname mydblib clear;
NOTE: Libref MYDBLIB has been deassigned.

```

**Example 4: Use Brief SQL Trace ',,,db'**

```

options sastrace=',,,db' sastraceloc=saslog nostsuffix;
libname mydblib oracle user=myusr1 password=mypwd1 path=mysrv1;
data mydblib.employee1;
  set mydblib.employee;
run;

```

Output is written to the SAS log, as specified in the SASTRACELOC=SASLOG option.

**Output 14.5 SAS Log Output from the SASTRACE=',,,db' System Option**

```

ORACLE_23: Prepared: on connection 2
SELECT * FROM EMPLOYEE
19?
ORACLE_24: Prepared: on connection 3
SELECT * FROM EMPLOYEE1
NOTE: SAS variable labels, formats, and lengths are not written to DBMS
      tables.
ORACLE_25: Executed: on connection 4
CREATE TABLE EMPLOYEE1(NAME VARCHAR2 (20),ID NUMBER (5),CITY VARCHAR2
(15),SALARY NUMBER ,DEPT NUMBER (5))
ORACLE_26: Executed: on connection 2
SELECT statement ORACLE_23
ORACLE_27: Prepared: on connection 4
INSERT INTO EMPLOYEE1 (NAME, ID, CITY, SALARY, DEPT) VALUES
(:NAME, :ID, :CITY, :SALARY, :DEPT)
**NOTE**: ORACLE_27 on connection 4
The Execute statements associated with
this Insert statement are suppressed due to SASTRACE brief
setting-SASTRACE=',,,bd'. Remove the 'b' to obtain full trace.
NOTE: There were 17 observations read from the data set MYDBLIB.EMPLOYEE.

```

**Example 5: Use Time Trace ',,,s'**

```

options sastrace=',,,s' sastraceloc=saslog nostsuffix;
libname mydblib oracle user=myusr1 password=mypwd1 schema=bday_data;
data mydblib.snow_birthdays;
  set work.winter_birthdays;
run;
libname mydblib clear;

```

Output is written to the SAS log, as specified in the SASTRACELOC=SASLOG option.

**Output 14.6 SAS Log Output from the SASTRACE=',,,s' System Option**

```

118 options sastrace=',,,s' sastraceloc=saslog nostsuffix;
119 libname mydblib oracle user=myusr1 password=mypwd1 schema=bday_data;
NOTE: Libref MYDBLIB was successfully assigned as follows:
      Engine:          ORACLE
      Physical Name:  lupin
120 data mydblib.snow_birthdays;
121   set work.winter_birthdays;
122 run;
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
NOTE: There were 3 observations read from the data set WORK.WINTER_BIRTHDAYS.
NOTE: The data set MYDBLIB.SNOW_BIRTHDAYS has 3 observations and 3 variables.
Summary Statistics for ORACLE are:
Total SQL execution seconds were:           0.127079
Total SQL prepare seconds were:             0.004404
Total SQL row insert seconds were:         0.004735
Total seconds used by the ORACLE ACCESS engine were 0.141860
NOTE: DATA statement used (Total process time):
      real time        0.21 seconds
      cpu time        0.04 seconds
123 libname mydblib clear;
NOTE: Libref MYDBLIB has been deassigned.

```

## Example 6: Use Time All Trace ',,sa'

```
options sastrace=',,sa' sastraceloc=saslog nostsuffix;
libname mydblib oracle user=myusr1 password=mypwd1 schema=bday_data;
data mydblib.snow_birthdays;
  set work.winter_birthdays;
run;
libname mydblib clear;
```

Output is written to the SAS log, as specified in the SASTRACELOC=SASLOG option.

### *Output 14.7 SAS Log Output from the SASTRACE= ',,sa' System Option*

```
146 options sastrace=',,sa' sastraceloc=saslog nostsuffix;
147
148 libname mydblib oracle user=myusr1 password=mypwd1 path=lupin
  schema=bday_data insertbuff=1;
NOTE: Libref MYDBLIB was successfully assigned as follows:
      Engine:          ORACLE
      Physical Name:  lupin
149 data mydblib.snow_birthdays;
150   set work.winter_birthdays;
151 run;
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
ORACLE:  The insert time in seconds is 0.004120
ORACLE:  The insert time in seconds is 0.001056
ORACLE:  The insert time in seconds is 0.000988
NOTE: There were 3 observations read from the data set WORK.WINTER_BIRTHDAYS.
NOTE: The data set MYDBLIB.SNOW_BIRTHDAYS has 3 observations and 3 variables.
Summary Statistics for ORACLE are:
Total SQL execution seconds were:           0.130448
Total SQL prepare seconds were:              0.004525
Total SQL row insert seconds were:           0.006158
Total seconds used by the ORACLE ACCESS engine were 0.147355
NOTE: DATA statement used (Total process time):
      real time        0.20 seconds
      cpu time         0.00 seconds
152
153 libname mydblib clear;
NOTE: Libref MYDBLIB has been deassigned.
```

## Example 7: Use Threaded Trace ',,t,'

```
options sastrace=',,t,' sastraceloc=saslog nostsuffix;
libname mydblib oracle user=myusr1 password=mypwd1 schema=bday_data;
data mydblib.snow_birthdays(DBTYPE=(empid'number(10'));
  set work.winter_birthdays;
run;
proc print data=mydblib.snow_birthdays(dbsliceparm=(all,3));
run;
```

Output is written to the SAS log, as specified in the SASTRACELOC=SASLOG option.

**Output 14.8 SAS Log Output from the SASTRACE= ',t,' System Option**

```

165 options sastrace=',,t,' sastraceloc=saslog nostsuffix;
166 data mydblib.snow_birthdays(DBTYPE=(empid='number(10)'));
167   set work.winter_birthdays;
168 run;
NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.
NOTE: There were 3 observations read from the data set WORK.WINTER_BIRTHDAYS.
NOTE: The data set MYDBLIB.SNOW_BIRTHDAYS has 3 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time          0.21 seconds
      cpu time           0.06 seconds
169 proc print data=mydblib.snow_birthdays(dbsliceparm=(all,3));
170 run;
ORACLE: DBSLICEPARM option set and 3 threads were requested
ORACLE: No application input on number of threads.
ORACLE: Thread 1 contains 1 obs.
ORACLE: Thread 2 contains 0 obs.
ORACLE: Thread 3 contains 2 obs.
ORACLE: Threaded read enabled. Number of threads created: 3
NOTE: There were 3 observations read from the data set MYDBLIB.SNOW_BaaaaaAYS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          1.12 seconds
      cpu time           0.17 seconds

```

## SASTRACELOC= SAS System Option

Specifies the location where SASTRACE= information should be printed.

|              |                                                                                                                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valid in:    | configuration file, SAS invocation, OPTIONS statement                                                                                                                                                                                                              |
| Category:    | Log and Procedure Output Control: SAS Log                                                                                                                                                                                                                          |
| Default:     | stdout                                                                                                                                                                                                                                                             |
| Data source: | Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Hadoop, HAWQ, Impala, Informix, JDBC, Microsoft SQL Server, MySQL, Netezza, ODBC, OLE DB, Oracle, PostgreSQL, SAP ASE, SAP HANA, SAP IQ, Snowflake, Spark, Teradata, Vertica |
| Notes:       | Support for Google BigQuery and Snowflake was added in the August 2019 release of SAS/ACCESS<br>Support for Spark was added in SAS 9.4M7.                                                                                                                          |
| See:         | <a href="#">SASTRACE= system option</a>                                                                                                                                                                                                                            |

## Syntax

**SASTRACELOC=**[stdout | SASLOG | FILE 'path-and-file-name'](#)

## Required Arguments

### **stdout**

specifies the default output location for your operating environment.

**SASLOG**

specifies that trace messages should be printed to the SAS log.

**FILE='path-and-file-name'**

specifies that trace messages should be printed to a file name that you provide. If you do not provide a path, the file is generated in the current working directory.

## Details

**SASTRACELOC=** lets you specify where to put the trace messages that **SASTRACE=** generates.

This option and its values might differ for each host.

## Example: Specify a Location for the Trace Log

This example writes trace information to the TRACE.LOG file in the work directory on the C drive on a PC platform.

```
options sastrace=',,,d' sastraceloc=file 'c:\work\trace.log';
```

## SQLGENERATION= SAS System Option

Specifies whether and when SAS procedures generate SQL for in-database processing of source data.

Valid in: SAS 9.4: configuration file, SAS invocation, OPTIONS statement, SAS System Options window, SAS Viya 3.5: configuration file, SAS invocation, OPTIONS statement, SASV9\_OPTIONS environment variable

Categories: Data Access  
System Administration: Performance

Default: SAS 9.4: (NONE DBMS='TERADATA DB2 ORACLE NETEZZA ASTER GREENPLM HADOOP SAPHANA IMPALA HAWQ POSTGRES REDSHIFT SQLSVR VERTICA BIGQUERY SNOW YBRICK SPARK MYSQL')

Restrictions: Parentheses are required when this option value contains multiple keywords.  
The maximum length of the option value is 4096 characters.

For DBMS= and EXCLUDEDDB= values, the maximum length of an engine name is eight characters. For the EXCLUDEPROC= value, the maximum length of a procedure name is 16 characters. An engine can appear only once, and a procedure can appear only once for a given engine.

Not all procedures support SQL generation for in-database processing for every engine type. If you specify a value that is not supported, an error message indicates the level of SQL generation that is not supported. The procedure can then reset to the default so that source table records can be read and processed within SAS. If this is not possible, the procedure ends and sets SYSERR= as needed.

If you are using the Metadata LIBNAME Engine, the only valid SQLGENERATION= modifiers are NONE and DBMS. The engine ignores the DBMS=, EXCLUDEDB=, and EXCLUDEPROC= modifiers.

- Requirement: You must specify NONE or DBMS as the primary state.
- Interactions: Use this option with such procedures as PROC FREQ to indicate that SQL is generated for in-database processing of DBMS tables through supported SAS/ACCESS engines. You can specify different SQLGENERATION= values for the DATA= and OUT= data sets by using different LIBNAME statements for each of these data sets.
- Data source: Amazon Redshift, Aster, DB2 under UNIX and PC Hosts, DB2 under z/OS, Google BigQuery, Greenplum, Hadoop, HAWQ, Impala, Microsoft SQL Server, MySQL, Netezza, Oracle, PostgreSQL, SAP HANA, Snowflake, Spark, Teradata, Vertica, Yellowbrick
- Notes: Support for Impala and HAWQ was added in SAS 9.4M3.  
Support for Google BigQuery and Yellowbrick was added in SAS 9.4M7.  
Support for MySQL was added in SAS 9.4M8.
- Tip: After you specify a required value (primary state), you can specify optional values (modifiers).
- See: [SQLGENERATION= LIBNAME option](#) (includes examples)  
“Running In-Database Procedures” in *SAS In-Database Products: User’s Guide*

## Syntax

```
SQLGENERATION=<(>NONE | DBMS  
<DBMS='engine1 engine2 ...engineN'>  
<EXCLUDEDB='engine | 'engine1...engineN'>  
<EXCLUDEPROC="engine='proc1 ... procN'...engineN='proc1 ... procN'"> <>  
SQLGENERATION=''
```

## Required Arguments

### NONE

prevents those SAS procedures that are enabled for in-database processing from generating SQL for in-database processing. This is a primary state.

### DBMS

allows SAS procedures that are enabled for in-database processing to generate SQL for in-database processing of DBMS tables through supported SAS/ACCESS engines. This is a primary state.

**Note:** As a best practice, run as many calculations in-database as possible. Processing that is run in-database generally results in better performance.

resets the value to the default that was shipped.

## Optional Arguments

**DBMS='engine1...engineN'**

specifies one or more SAS/ACCESS engines. It modifies the primary state.

**EXCLUDEDB=engine | 'engine1...engineN'**

prevents SAS procedures from generating SQL for in-database processing for one or more specified SAS/ACCESS engines.

**EXCLUDEPROC="engine='proc1...procN' ...engineN='proc1...procN' "**

prevents one or more specified SAS procedures from generating SQL for in-database processing for one or more specified SAS/ACCESS engines.

---

## Details

Here are the values that you specify for each engine. The values are not case-specific.

*Table 14.2 SQLGENERATION Values to Specify for Each Engine*

| Engine               | SQLGENERATION Value |
|----------------------|---------------------|
| Amazon Redshift      | REDSHIFT            |
| Aster                | ASTER               |
| DB2                  | DB2                 |
| Google BigQuery      | BIGQUERY            |
| Greenplum            | GREENPLUM           |
| Hadoop               | HADOOP              |
| HAWQ                 | HAWQ                |
| Impala               | IMPALA              |
| Microsoft SQL Server | SQLSVR              |
| MySQL                | MYSQL               |
| Netezza              | NETEZZA             |
| Oracle               | ORACLE              |
| PostgreSQL           | POSTGRES            |
| SAP HANA             | SAPHANA             |

| Engine      | SQLGENERATION Value |
|-------------|---------------------|
| Snowflake   | SNOW                |
| Teradata    | TERADATA            |
| Vertica     | VERTICA             |
| Yellowbrick | YBRICK              |

Here is how SAS/ACCESS handles precedence between the LIBNAME and system option.

**Table 14.3** Precedence of Values for SQLGENERATION= LIBNAME and System Options

| LIBNAME Option | PROC EXCLUDE on System Option? | Engine Specified on System Option | Resulting Value | From (option) |
|----------------|--------------------------------|-----------------------------------|-----------------|---------------|
| not specified  | yes                            | NONE                              | NONE            | system        |
|                |                                | DBMS                              | EXCLUDEPROC     |               |
| NONE           |                                | NONE                              | NONE            | LIBNAME       |
|                |                                | DBMS                              |                 |               |
| DBMS           |                                | NONE                              | EXCLUDEPROC     | system        |
|                |                                | DBMS                              |                 |               |
| not specified  | no                             | NONE                              | NONE            |               |
|                |                                | DBMS                              | DBMS            |               |
| NONE           |                                | NONE                              | NONE            | LIBNAME       |
|                |                                | DBMS                              |                 |               |
| DBMS           |                                | NONE                              | DBMS            |               |
|                |                                | DBMS                              |                 |               |

## Example

Here is the default that is shipped with the product.

```
options sqlgeneration='';
proc options option=sqlgeneration;
run;
```

SAS procedures generate SQL for in-database processing for all databases except DB2 in this example.

```
options sqlgeneration='';
options sqlgeneration=(DBMS EXCLUDEDDB='DB2');
proc options option=sqlgeneration;
run;
```

In this example, in-database processing occurs only for Teradata. SAS procedures that are run on other databases do not generate SQL for in-database processing.

```
options sqlgeneration='';
options SQLGENERATION=(NONE DBMS='Teradata');
proc options option=sqlgeneration;
run;
```

For this example, SAS procedures generate SQL for Teradata and Oracle in-database processing. However, no SQL is generated for PROC1 and PROC2 in Oracle.

```
options sqlgeneration='';
options SQLGENERATION = (NONE DBMS='Teradata Oracle'
    EXCLUDEPROC="oracle='proc1 proc2'");
proc options option=sqlgeneration;
run;
```

## SQLMAPPUTTO= SAS System Option

Specifies whether the PUT function is mapped to the SAS\_PUT function for a database. Use this system option to specify an alternative database in which the SAS\_PUT function is published.

Valid in: configuration file, SAS invocation, OPTIONS statement

Category: Files: SAS Files

Default: SAS\_PUT

Data source: Aster, DB2 under UNIX and PC Hosts, Greenplum, Netezza, Teradata

See: [SQL\\_FUNCTIONS= LIBNAME option](#), *SAS In-Database Products: User's Guide*

## Syntax

**SQLMAPPUTTO=NONE | SAS\_PUT | (database.SAS\_PUT)**

## Required Arguments

### **NONE**

specifies to PROC SQL that no PUT function mapping is to occur.

### **SAS\_PUT**

specifies that the PUT function be mapped to the SAS\_PUT function in the database.

**database.SAS\_PUT**

specifies the database name where the SAS\_PUT function and format definitions reside.

**TIP** It is not necessary that the format definitions and the SAS\_PUT function reside in the same database as the one that contains the data that you want to format. Use the *database.SAS\_PUT* argument to specify the database where the format definitions and the SAS\_PUT function are published.

**TIP** The database name can be a multilevel name and it can include blanks.

**Requirement** If you specify a database name, you must enclose the entire argument in parentheses.

---

## Details

SAS format publishing macros publish the PUT function implementation to the database as a new function named SAS\_PUT. The format publishing macros also publish both user-defined formats that you create using PROC FORMAT and formats that SAS supplies. The SAS\_PUT function supports the use of SAS formats. You can use it in SQL queries that SAS submits to the database so that the entire SQL query can be processed inside the database. You can also use it in conjunction with in-database procedures.

You can use this option with the SQLREDUCEPUT=, SQLREDUCEPUTOBS, and SQLREDUCEPUTVALUES= system options. For more information about these options, see *SAS SQL Procedure User's Guide*.

---

## VALIDVARNAME= SAS System Option

Controls the type of SAS variable names that can be used or created during a SAS session.

|           |                                                                                        |
|-----------|----------------------------------------------------------------------------------------|
| Valid in: | configuration file, SAS invocation, OPTIONS statement, SAS System Options window       |
| Category: | Files: SAS Files                                                                       |
| Default:  | V7                                                                                     |
| See:      | <a href="#">Introduction to SAS/ACCESS Naming</a> , <b>VALIDMEMNAME=</b> system option |

---

## Syntax

**VALIDVARNAME=**[V6 | V7 | UPCASE | ANY](#)

## Required Arguments

### **VALIDVARNAME=V6**

indicates that a DBMS column name is changed to a valid SAS name, following these rules:

- Up to eight alphanumeric characters are allowed. Names that are longer than eight characters are truncated. If required, numbers are appended to the ends of the truncated names to make them unique.
- Mixed-case characters are converted to uppercase.
- Special characters are not allowed.

---

**Note:** The primary reason for using this value is for existing SAS code that references variables that use older naming rules.

---

### **VALIDVARNAME=V7**

indicates that a DBMS column name is changed to a valid SAS name, following these rules. This is the default value for SAS 7 and later.

- Up to 32 mixed-case alphanumeric characters are allowed.
- Names must begin with an alphabetic character or an underscore.
- Invalid characters are changed to underscores.
- Any column name that is not unique when it is normalized is made unique by appending a counter (0, 1, 2, and so on) to the name.

### **VALIDVARNAME=UPCASE**

indicates that a DBMS column name is changed to a valid SAS name as described in VALIDVARNAME=V7 except that variable names are in uppercase.

### **VALIDVARNAME=ANY**

allows any characters in DBMS column names to appear as valid characters in SAS variable names. Symbols, such as the equal sign (=) and the asterisk (\*), must be contained in a '*variable-name*' construct. You must use ANY whenever you want to read DBMS column names that do not follow the SAS naming conventions.

Up to 32 characters are allowed in a column name.

Any column name that is not unique when it is normalized is made unique by appending a count (0, 1, 2, and so on).

---

## Details

The VALIDVARNAME= system option is supported for all DBMSs that support the SQL pass-through facility. You can set this option on start-up or in an OPTIONS statement, and the option value is used in the call to the SQL procedure.

Alternatively, you can specify the VALIDVARNAME= option in the CONNECT statement.

The VALIDVARNAME= system option is often used with the VALIDMEMNAME= system option. For more information, see “[VALIDMEMNAME= System Option](#)” in *SAS System Options: Reference*.

## Examples

### Example 1: Rename Columns during View Creation

This example shows how the SQL pass-through facility works with VALIDVARNAME=V6.

```
options validvarname=v6;
proc sql;
    connect to hadoop (user=myusr1 pass=mypwd1);
    create view myview as
        select amount_b, amount_s
        from connection to hadoop
            (select "Amount Budgeted$", "Amount Spent$"
            from mytable);
    quit;
proc contents data=myview;
run;
```

Output from this example would show that "Amount Budgeted\$" becomes AMOUNT\_B and "Amount Spent\$" becomes AMOUNT\_S.

### Example 2: Pass VALIDVARNAME= as a Connection Option

This example shows how you can pass VALIDVARNAME= as a connection option in the CONNECT statement in the SQL procedure.

```
proc sql;
    drop view work.TLV1;
    connect to hadoop ( validvarname=v7 server="myserver" port=5433
    user=myuserid password=mypwd database=test );
        exec("drop table vartb") by hadoop;
        exec("create table vartb ( c1234567890 int,
    c123456789012345678 float ) ") by hadoop;
        exec("insert into vartb values (123, 3.14159)") by hadoop;

    create view tlv1 as
        select * from connection to hadoop (select * from vartb);
quit;
```

# SQL Pass-Through Facility

---

|                                                                         |            |
|-------------------------------------------------------------------------|------------|
| <b>About SQL Procedure Interactions .....</b>                           | <b>689</b> |
| Overview of SQL Procedure Interactions with SAS/ACCESS .....            | 689        |
| SQL Pass-Through Facility .....                                         | 689        |
| <b>Syntax: SQL Pass-Through Facility for Relational Databases .....</b> | <b>690</b> |
| Overview .....                                                          | 690        |
| Return Codes .....                                                      | 690        |
| <b>Dictionary .....</b>                                                 | <b>691</b> |
| CONNECT Statement .....                                                 | 691        |
| CONNECTION TO Component .....                                           | 696        |
| DISCONNECT Statement .....                                              | 699        |
| EXECUTE Statement .....                                                 | 700        |

---

## About SQL Procedure Interactions

---

### Overview of SQL Procedure Interactions with SAS/ACCESS

The SQL procedure implements Structured Query Language (SQL) for SAS software. For details about PROC SQL , see the [SAS SQL Procedure User's Guide](#). Here is how you can use SAS/ACCESS software for relational databases for PROC SQL interactions.

- You can assign a libref to a DBMS using the SAS/ACCESS LIBNAME statement and reference the new libref in a PROC SQL statement to query, update, or delete DBMS data. (See [LIBNAME Statement for Relational Databases](#) on page [127](#).)
- You can embed LIBNAME information in a PROC SQL view and then automatically connect to the DBMS every time the PROC SQL view is

- processed. (See [SQL Views with Embedded LIBNAME Statements on page 132](#).)
  - You can send DBMS-specific SQL statements directly to a DBMS using an extension to PROC SQL called the SQL pass-through facility. (See [Syntax for the SQL pass-through facility for Relational Databases on page 690](#).)
- 

## SQL Pass-Through Facility

The SQL pass-through facility uses SAS/ACCESS to connect to a DBMS and to send statements directly to the DBMS for execution. As an alternative to the SAS/ACCESS LIBNAME statement, this facility lets you use the SQL syntax of your DBMS. It supports any SQL that is not ANSI-standard that your DBMS supports.

Not all SAS/ACCESS interfaces support this feature, however. To determine whether it is available in your environment, see [SAS/ACCESS features by host on page 96](#).

Here are the tasks that you can complete by using the SQL pass-through facility.

- Establish and terminate connections with a DBMS using its [CONNECT](#) and [DISCONNECT](#) statements.
- Send dynamic, non-query, DBMS-specific SQL statements to a DBMS using its [EXECUTE](#) statement.
- Retrieve data directly from a DBMS using its [CONNECTION TO](#) component in the FROM clause of a PROC SQL SELECT statement.

You can use SQL pass-through facility statements in a PROC SQL query, or you can store them in an SQL view. When you create an SQL view, any arguments that you specify in the CONNECT statement are stored with the view. Therefore, when you use the view in a SAS program, SAS can establish the appropriate connection to the DBMS.

For DBMS-specific details about the SQL pass-through facility, see the reference section for your SAS/ACCESS interface.

---

## Syntax: SQL Pass-Through Facility for Relational Databases

---

### Overview

The [syntax](#) section presents the syntax for the SQL pass-through facility statements and the CONNECTION TO component. For DBMS-specific details, see the DBMS-specific reference section for your SAS/ACCESS interface.

**PROC SQL <options>;**

---

```
CONNECT TO dbms-name <AS alias>
<<database-connection-arguments> <connect-statement-arguments> )>;
DISCONNECT FROM dbms-name | alias;
EXECUTE (dbms-specific-SQL-statement) BY dbms-name | alias;
SELECT column-list FROM CONNECTION TO dbms-name | alias (dbms-query)
```

---

## Return Codes

As you use the PROC SQL statements that are available in the SQL pass-through facility, any error return codes and error messages are written to the SAS log. These codes and messages are available to you through these SAS macro variables:

SQLXRC

contains the DBMS return code that identifies the DBMS error.

SQLXMSG

contains descriptive information about the DBMS error that the DBMS generates.

The contents of the SQLXRC and SQLXMSG macro variables are printed in the SAS log using the %PUT macro. They are reset after each SQL pass-through facility statement has been executed. For details about these return codes, see “Macro Variables for Relational Databases” on page 656.

---

## Dictionary

---

## CONNECT Statement

Establishes a connection with a DBMS.

Valid in: PROC SQL steps (when accessing DBMS data using SAS/ACCESS software)

---

## Syntax

```
CONNECT TO dbms-name < AS alias >
< (connect-statement-arguments) >
< (database-connection-arguments ) >;
CONNECT USING libref < AS alias >;
```

## Required Arguments

### ***dbms-name***

specifies the DBMS to which you want to connect. You must specify the DBMS name for your SAS/ACCESS interface. See the SQL pass-through section in the DBMS-specific reference section for your SAS/ACCESS interface.

### ***libref***

specifies the libref for which a DBMS connection has already been established through the LIBNAME statement.

## Optional Arguments

### **AS *alias***

specifies an alias for the connection that has 1 to 32 characters. The AS keyword must precede *alias*. If you do not specify an alias, the DBMS name is used as the name of the SQL pass-through connection. Some DBMSs allow more than one connection. You can use the AS clause to name connections so that you can refer to them later.

### ***connect-statement-arguments***

specifies values for arguments that indicate whether you can make multiple connections, shared or unique connections, and so on, to the database. With these arguments, the SQL pass-through facility can use some of the connection management features of the LIBNAME statement or of SAS system options. Although these arguments are optional, you must enclose them in parentheses if you include any:

- [CONNECTION=](#)
- [CONNECTION\\_GROUP=](#)
- [DBCONINIT=](#)
- [DBCONTERM=](#)
- [DBGEN\\_NAME=](#)
- [DBMAX\\_TEXT=](#)
- [DBPROMPT=](#)
- [DEFER=](#)
- [VALIDVARNAME=](#)

---

**Note:** In addition to the arguments that are listed here, several other LIBNAME or system options are available for use with the CONNECT statement. For options that are available for your SAS/ACCESS interface, see the SQL pass-through facility section for your interface in the DBMS-specific reference section. When used with the SQL pass-through facility CONNECT statement, these options have the same effect as they do in a LIBNAME statement.

---

**CONNECTION= SHARED | GLOBAL**

indicates that multiple CONNECT statements for a DBMS can use the same connection.

The CONNECTION= option enables you to control the number of connections, and therefore transactions, that your SAS/ACCESS engine executes and supports for each SQL pass-through CONNECT statement.

When CONNECTION=GLOBAL, multiple CONNECT statements that use identical values for CONNECTION=, CONNECTION\_GROUP=, DBCONINIT=, DBCONTERM=, and any database connection arguments can share the same connection to the DBMS. For more information, see ["Conditions for a Shared DBMS Connection" on page 185](#).

When CONNECTION=SHARED, the CONNECT statement makes one connection to the DBMS. Only SQL pass-through statements that use this alias share the connection. SHARED is the default value for CONNECTION=.

In this example, the two CONNECT statements share the same connection to the DBMS because CONNECTION=GLOBAL. Only the first CONNECT statement actually makes the connection to the DBMS. The last DISCONNECT statement is the only statement that disconnects from the DBMS.

```
proc sql;
/*...SQL Pass-Through statements referring to mybone...*/
connect to oracle as mybone
  (user=myusr1 pw=mypwd1 path='mysrv1' connection=global);
/*...SQL Pass-Through statements referring to mydbtwo...*/
connect to oracle as mydbtwo
  (user=myusr1 pw=mypwd1 path='mysrv1' connection=global);
disconnect from mybone;
disconnect from mydbtwo;
quit;
```

**CONNECTION\_GROUP=***connection-group-name*  
specifies a connection that can be shared among several CONNECT statements in the SQL pass-through facility.

*Default:* none

By specifying the name of a connection group, you can share one DBMS connection among several CONNECT statements. The connection to the DBMS can be shared only if each CONNECT statement specifies the same CONNECTION\_GROUP= value and specifies identical DBMS connection arguments. For more information, see ["Conditions for a Shared DBMS Connection" on page 185](#).

When CONNECTION\_GROUP= is specified, it implies that the value of the CONNECTION= option is GLOBAL.

**DBCONINIT=<'>DBMS-user-command<'>**  
specifies a user-defined initialization command to be executed immediately after the connection to the DBMS.

You can specify any DBMS command that can be passed by the SAS/ACCESS engine to the DBMS and that does not return a result set or output parameters. The command executes immediately after the DBMS connection is established successfully. If the command fails, a disconnect occurs, and the CONNECT statement fails. You must specify the command as a single, quoted string, unless it is an environment variable.

**DBCONTERM='DBMS-user-command'**  
specifies a user-defined termination command to be executed before the disconnect from the DBMS that occurs with the DISCONNECT statement.

*Default:* none

The termination command that you select can be a script, stored procedure, or any DBMS SQL language statement that might provide additional control over the interaction between the SAS/ACCESS engine and the DBMS. You can specify any valid DBMS command that can be passed by the SAS/ACCESS engine to the DBMS and that does not return a result set or output parameters. The command executes immediately before SAS terminates each connection to the DBMS. If the command fails, SAS provides a warning message but the disconnect still occurs. You must specify the command as a quoted string.

**DBGEN\_NAME=** DBMS | SAS

specifies whether to automatically rename DBMS columns containing characters that SAS does not allow, such as \$, to valid SAS variable names.

**DBMAX\_TEXT=***integer*

determines the length of any very long DBMS character data type that is read into SAS or written from SAS when using a SAS/ACCESS engine. This option applies to reading, appending, and updating rows in an existing table. It does not apply when you are creating a table.

Examples of a long DBMS data type are the SAP ASE TEXT data type and the Oracle LONG RAW data type.

**DBPROMPT=YES | NO**

specifies whether SAS displays a window that prompts the user to enter DBMS connection information before connecting to the DBMS.

*Default:* NO

*Interaction:* DEFER= LIBNAME option

If you specify DBPROMPT=YES, SAS displays a window that interactively prompts you for the DBMS connection arguments when the CONNECT statement is executed. Therefore, it is not necessary to provide connection arguments with the CONNECT statement. If you do specify connection arguments with the CONNECT statement and you specify DBPROMPT=YES, the connection argument values are displayed in the window. These values can be overridden interactively.

If you specify DBPROMPT=NO, SAS does not display the prompting window.

The DBPROMPT= option interacts with the DEFER= LIBNAME option to determine when the prompt window appears. If DEFER=NO, the DBPROMPT window appears when the CONNECT statement is executed. If DEFER=YES, the DBPROMPT window appears the first time a pass-through statement is executed. The DEFER= option normally defaults to NO. The option defaults to YES if DBPROMPT=YES. You can override this default by explicitly specifying DEFER=NO.

**DEFER=NO | YES**

determines when the connection to the DBMS occurs.

*Default:* NO

If DEFER=YES, the connection to the DBMS occurs when the first SQL pass-through statement is executed. If DEFER=NO, the connection to the DBMS occurs when the CONNECT statement occurs.

**VALIDVARNAME=V6 | V7 | ANY**

indicates the compatibility mode for variable names for the SQL pass-through facility. The default value is V7. For more information about the V7 or ANY values, see “[VALIDVARNAME= SAS System Option](#)” on page 686.

By default, DBMS column names are changed to valid SAS names, following these rules:

- Up to 32 mixed-case alphanumeric characters are allowed.
- Names must begin with an alphabetic character or an underscore.
- Characters that are not permitted are changed to underscores.
- Any column name that is not unique when it is normalized is made unique by appending a counter (0,1,2,...) to the name.

When VALIDVARNAME=V6 is specified, the SAS/ACCESS engine for the DBMS truncates column names to eight characters, as it does in SAS 6. If required, numbers are appended to the ends of the truncated names to make them unique. Specifying this option overrides the value of the SAS system option VALIDVARNAME= during (and only during) the SQL pass-through connection.

This example shows how the SQL pass-through facility uses VALIDVARNAME=V6 as a connection argument. Using this option causes the output to show the DBMS column "Amount Budgeted\$" as AMOUNT\_B and "Amount Spent\$" as AMOUNT\_S.

```
proc sql;
connect to netezza (server='myserver' database='myDB' user=myusr1
                     password=mypwd1 validvarname=v6)
create table work.foo as
  select * from connection to netezza
    (select "Amount Budgeted$", "Amount Spent$"
     from annual_budget);
quit;

proc contents data=work.foo;run;
```

For this example, if you omit VALIDVARNAME=V6 as a connection argument, you must add it in an OPTIONS= statement in order for PROC CONTENTS to work.

```
options validvarname=v6;
proc contents data=work.foo; run;
```

The primary reason for using this option would be in a SAS session that is running with option VALIDVARNAME=V7, but in which you have existing SAS code that references variables that use older naming rules.

***database-connection-arguments***

specifies values for DBMS-specific arguments that PROC SQL needs to connect to the DBMS. Though they are optional for most databases, you must enclose them in parentheses if you include any. For information about these arguments, see the DBMS-specific reference section for your SAS/ACCESS interface.

## Details

The CONNECT statement establishes a connection with the DBMS. You establish a connection to send DBMS-specific SQL statements to the DBMS or to retrieve DBMS data. The connection remains in effect until you issue a [DISCONNECT](#) statement or terminate the SQL procedure.

Follow these steps to connect to a DBMS using the SQL pass-through facility.

- 1 Initiate a PROC SQL step.
- 2 Use the SQL pass-through facility CONNECT statement, identify the DBMS (such as Hadoop, Oracle, or DB2), and assign an (optional) alias.
- 3 Specify any attributes for the connection such as SHARED or UNIQUE.
- 4 Specify any arguments that are needed to connect to the database.

The CONNECT statement is optional for some DBMSs. However, if you do not specify it, the default values for all database connection arguments are used.

A CONNECT USING statement uses the existing connection that is established in the LIBNAME statement for the specified libref.

Any return code or message that the DBMS generates is available in the SQLXRC and SQLXMSG macro variables after the statement executes. For information about these macro variables, see [“Macro Variables for Relational Databases” on page 656](#).

## Example: CONNECT Statement

This example connects to an SAP ASE server and assigns the alias ASECON1 to it. SAP ASE is a case-sensitive database, so database objects are in uppercase, as they were created.

```
proc sql;
  connect to sapase as asecon1
    (server=MYSRV1 database=PERSONNEL user=MYUSR1
     password=MYPWD1 connection=global);
  %put &sqlxmsg &sqlxrc;
```

You might be able to omit the CONNECT statement and implicitly connect to a database by using default values. For details, see the DBMS-specific reference section for your SAS/ACCESS interface.

## CONNECTION TO Component

Retrieves and uses DBMS data in a PROC SQL query or view

|           |                                                                                      |
|-----------|--------------------------------------------------------------------------------------|
| Valid in: | PROC SQL step SELECT statements (when accessing DBMS data using SAS/ACCESS software) |
|-----------|--------------------------------------------------------------------------------------|

# Syntax

**CONNECTION TO** *dbms-name* | *alias* | *(dbms-query)*

## Required Arguments

### *dbms-name*

identifies the DBMS to which you direct the DBMS-specific SQL statement. See the SQL pass-through section in the DBMS-specific reference section for your SAS/ACCESS interface.

### *alias*

specifies an alias, if one was specified in the CONNECT statement.

### *(dbms-query)*

specifies the query that you are sending to the DBMS. The query can use any DBMS-specific SQL statement or syntax that is valid for the DBMS.

You must specify a query argument in the CONNECTION TO component, and the query must be enclosed in parentheses. The query is passed to the DBMS exactly as you enter it. Therefore, if your DBMS is case sensitive, you must use the correct case for DBMS object names.

On some DBMSs, the *dbms-query* argument can be a DBMS stored procedure. However, stored procedures with output parameters are not supported in the SQL pass-through facility. Furthermore, if the stored procedure contains more than one query, only the first query is processed.

---

## Details

The CONNECTION TO component specifies the DBMS connection that you want to use or that you want to create. Specify the CONNECTION TO component if you have omitted the CONNECT statement. CONNECTION TO then enables you to retrieve DBMS data directly through a PROC SQL query.

You use the CONNECTION TO component in the FROM clause of a PROC SQL SELECT statement:

```
PROC SQL;
  SELECT column-list
    FROM CONNECTION TO dbms-name | (dbms-query) | other optional PROC SQL clauses
  QUIT;
```

You can use CONNECTION TO in any FROM clause, including those in nested queries—that is, subqueries.

You can store an SQL pass-through facility query in an SQL view and then use that view in SAS programs. When you create an SQL view, any options that you specify in the corresponding CONNECT statement are stored too. So when the SQL view is used in a SAS program, SAS can establish the appropriate connection to the DBMS.

On many relational databases, you can issue a CONNECTION TO component in a PROC SQL SELECT statement directly without first connecting to a DBMS. (See [CONNECTION statement on page 691](#).) If you omit the CONNECT statement, an

implicit connection is performed by using default values for all database connection arguments. This automatic connection occurs at the first PROC SQL SELECT statement that contains a CONNECTION TO component. For details, see the SQL pass-through section in the DBMS-specific reference section for your SAS/ACCESS interface.

Because relational databases and SAS have different naming conventions, some DBMS column names might be changed when you retrieve DBMS data through the CONNECTION TO component. See [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#) for more information.

## Examples

### Example 1: Send an Oracle SQL Query to the Oracle Database

After you connect to a DBMS by using either the CONNECT statement or implicit default values, you can send a DBMS-specific SQL query to the DBMS using the CONNECTION TO component. You specify the columns that you want to retrieve, identify your DBMS type, and issue your query by using the SQL syntax of your DBMS.

This example sends an Oracle SQL query (highlighted below) to the Oracle database for processing. The results from the Oracle SQL query serve as a virtual table for the PROC SQL FROM clause. In this example, MYCON is a connection alias.

```
proc sql;
  connect to oracle as mycon (user=myusr1
                                password=mypwd1 path='mysrv1');
  %put &sqlxmsg;
  select *
    from connection to mycon
      (select empid, lastname, firstname,
             hiredate, salary
       from employees where
             hiredate>='31-DEC-88');
  %put &sqlxmsg;
  disconnect from mycon;
  quit;
```

The SAS %PUT macro displays the &SQLXMSG macro variable for error codes and information from the DBMS. For details, see [“Macro Variables for Relational Databases” on page 656](#).

### Example 2: Name and Store the Query as an SQL View

This example gives the query a name and stores it as the SQL view samples.

```
libname samples 'SAS-library';
proc sql;
  connect to oracle as mycon (user=myusr1
```

```
password=mypwd1 path='mysrv1');
%put &sqlxmsg;
create view samples.hires88 as
  select *
    from connection to mycon
      (select empid, lastname, firstname,
             hiredate, salary
      from employees where
             hiredate>='31-DEC-88');
%put &sqlxmsg;
disconnect from mycon;
quit;
```

---

## DISCONNECT Statement

Terminates the connection to a DBMS.

Valid in: PROC SQL steps (when accessing DBMS data using SAS/ACCESS software)

---

## Syntax

**DISCONNECT FROM *dbms-name* | *alias***

### Required Arguments

***dbms-name***

specifies the DBMS from which you want to disconnect. You must either specify the DBMS name for your SAS/ACCESS interface or use an alias in this statement. See the LIBNAME section in the DBMS-specific reference section for your SAS/ACCESS interface. If you used the [CONNECT](#) statement to connect to the DBMS, the DBMS name or alias in the DISCONNECT statement must match what you specified in the CONNECT statement.

***alias***

specifies an alias that was defined in the CONNECT statement.

---

## Details

The DISCONNECT statement ends the connection with the DBMS. If you do not include the DISCONNECT statement, SAS performs an implicit DISCONNECT when PROC SQL terminates. The SQL procedure continues to execute until you submit a QUIT statement, another SAS procedure, or a DATA step.

Any return code or message that is generated by the DBMS is available in the macro variables SQLRC and SQLXMSG after the statement executes. See “[Macro Variables for Relational Databases](#)” on page 656 for more information about these macro variables.

## Example

To exit the SQL pass-through facility, use the facilities DISCONNECT statement and then QUIT the PROC SQL statement. This example disconnects the user from a DB2 database with the alias DBCON1 and terminates the SQL procedure.

```
proc sql;
  connect to db2 as dbcon1 (ssid=db2a);
  ...more SAS statements...
  disconnect from dbcon1;
quit;
```

---

## EXECUTE Statement

Sends DBMS-specific, non-query SQL statements to the DBMS.

Valid in: PROC SQL steps (when accessing DBMS data using SAS/ACCESS software)

---

## Syntax

**EXECUTE** (*dbms-specific-sql-statement*) BY *dbms-name* | *alias*;

### Required Arguments

**(*dbms-specific-sql-statement*)**

a dynamic non-query, DBMS-specific SQL statement. This argument is required and must be enclosed in parentheses. The SQL statement might be case sensitive, depending on your DBMS, and it is passed to the DBMS exactly as you enter it.

On some DBMSs, this argument can be a DBMS stored procedure. However, stored procedures with output parameters are not supported in the SQL pass-through facility. Furthermore, if the stored procedure contains more than one query, only the first query is processed.

Any return code or message that is generated by the DBMS is available in the macro variables SQLXRC and SQLXMSG after the statement executes. See [Macro Variables for Relational Databases on page 656](#) for more information about these macro variables.

***dbms-name***

specifies the DBMS to which you direct the DBMS-specific SQL statement. The keyword BY must appear before the *dbms-name* argument. You must either specify the DBMS name for your SAS/ACCESS interface or use an alias. See the SQL pass-through section in the DBMS-specific reference section for your SAS/ACCESS interface.

***alias***

specifies an alias that was defined in the [CONNECT](#) statement. (You cannot use an alias if the CONNECT statement was omitted.)

# Details

## Overview

The EXECUTE statement sends dynamic non-query, DBMS-specific SQL statements to the DBMS and processes those statements.

In some SAS/ACCESS interfaces, you can issue an EXECUTE statement directly without first explicitly connecting to a DBMS. (See [CONNECT statement on page 691](#).) If you omit the CONNECT statement, an implicit connection is performed by using default values for all database connection arguments when the first EXECUTE statement is passed to the DBMS. For details, see the SQL pass-through section in the DBMS-specific reference section for your SAS/ACCESS interface.

The EXECUTE statement cannot be stored as part of an SQL pass-through facility query in a PROC SQL view.

## Useful Statements to Include in EXECUTE Statements

You can pass these statements to the DBMS by using the SQL pass-through facility EXECUTE statement.

*Table 15.1 Statements That Can Be Passed to the DBMS*

| Statement | Description                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------|
| CREATE    | creates a DBMS table, view, index, or other DBMS object, depending on how the statement is specified. |
| DELETE    | deletes rows from a DBMS table.                                                                       |
| DROP      | deletes a DBMS table, view, or other DBMS object, depending on how the statement is specified.        |
| GRANT     | gives users the authority to access or modify objects such as tables or views.                        |
| INSERT    | adds rows to a DBMS table.                                                                            |
| REVOKE    | revokes the access or modification privileges that were given to users by the GRANT statement.        |
| UPDATE    | modifies the data in one column of a row in a DBMS table.                                             |

For more information and restrictions on these and other SQL statements, see the SQL documentation for your DBMS.



**PART 3****DBMS-Specific Reference**

|                   |                                                            |      |
|-------------------|------------------------------------------------------------|------|
| <i>Chapter 16</i> | <i>SAS/ACCESS Interface to Amazon Redshift</i>             | 705  |
| <i>Chapter 17</i> | <i>SAS/ACCESS Interface to Aster</i>                       | 727  |
| <i>Chapter 18</i> | <i>SAS/ACCESS Interface to DB2 under UNIX and PC Hosts</i> | 749  |
| <i>Chapter 19</i> | <i>SAS/ACCESS Interface to DB2 under z/OS</i>              | 785  |
| <i>Chapter 20</i> | <i>SAS/ACCESS Interface to Google BigQuery</i>             | 847  |
| <i>Chapter 21</i> | <i>SAS/ACCESS Interface to Greenplum</i>                   | 869  |
| <i>Chapter 22</i> | <i>SAS/ACCESS Interface to Hadoop</i>                      | 895  |
| <i>Chapter 23</i> | <i>SAS/ACCESS Interface to HAWQ</i>                        | 925  |
| <i>Chapter 24</i> | <i>SAS/ACCESS Interface to Impala</i>                      | 951  |
| <i>Chapter 25</i> | <i>SAS/ACCESS Interface to Informix</i>                    | 971  |
| <i>Chapter 26</i> | <i>SAS/ACCESS Interface to JDBC</i>                        | 991  |
| <i>Chapter 27</i> | <i>SAS/ACCESS Interface to Microsoft SQL Server</i>        | 1007 |
| <i>Chapter 28</i> | <i>SAS/ACCESS Interface to MySQL</i>                       | 1031 |
| <i>Chapter 29</i> | <i>SAS/ACCESS Interface to Netezza</i>                     | 1049 |

|                   |                                            |      |
|-------------------|--------------------------------------------|------|
| <i>Chapter 30</i> | <i>SAS/ACCESS Interface to ODBC</i>        | 1069 |
| <i>Chapter 31</i> | <i>SAS/ACCESS Interface to OLE DB</i>      | 1103 |
| <i>Chapter 32</i> | <i>SAS/ACCESS Interface to Oracle</i>      | 1131 |
| <i>Chapter 33</i> | <i>SAS/ACCESS Interface to PostgreSQL</i>  | 1171 |
| <i>Chapter 34</i> | <i>SAS/ACCESS Interface to SAP ASE</i>     | 1195 |
| <i>Chapter 35</i> | <i>SAS/ACCESS Interface to SAP HANA</i>    | 1223 |
| <i>Chapter 36</i> | <i>SAS/ACCESS Interface to SAP IQ</i>      | 1249 |
| <i>Chapter 37</i> | <i>SAS/ACCESS Interface to Snowflake</i>   | 1271 |
| <i>Chapter 38</i> | <i>SAS/ACCESS Interface to Spark</i>       | 1297 |
| <i>Chapter 39</i> | <i>SAS/ACCESS Interface to Teradata</i>    | 1319 |
| <i>Chapter 40</i> | <i>SAS/ACCESS Interface to Vertica</i>     | 1383 |
| <i>Chapter 41</i> | <i>SAS/ACCESS Interface to Yellowbrick</i> | 1399 |

# SAS/ACCESS Interface to Amazon Redshift

|                                                                        |     |
|------------------------------------------------------------------------|-----|
| <i>System Requirements for SAS/ACCESS Interface to Amazon Redshift</i> | 706 |
| <i>Introduction to SAS/ACCESS Interface to Amazon Redshift</i>         | 706 |
| <i>LIBNAME Statement for the Amazon Redshift Engine</i>                | 707 |
| Overview                                                               | 707 |
| Arguments                                                              | 707 |
| Best Practice: Specify the Schema                                      | 712 |
| Amazon Redshift LIBNAME Statement Examples                             | 712 |
| <i>Data Set Options for Amazon Redshift</i>                            | 713 |
| <i>SQL Pass-Through Facility Specifics for Amazon Redshift</i>         | 716 |
| Key Information                                                        | 716 |
| CONNECT Statement Examples for Amazon Redshift                         | 716 |
| <i>Passing SAS Functions to Amazon Redshift</i>                        | 717 |
| <i>Passing Joins to Amazon Redshift</i>                                | 718 |
| <i>Bulk Loading for Amazon Redshift</i>                                | 718 |
| Overview                                                               | 718 |
| Data Set Options with Bulk Loading                                     | 719 |
| LIBNAME Options for Bulk Unloading                                     | 719 |
| Examples of Bulk Loading                                               | 720 |
| <i>Encryption with Amazon Redshift</i>                                 | 721 |
| <i>Locking in the Amazon Redshift Interface</i>                        | 721 |
| <i>Working with NLS Characters in Amazon Redshift Data</i>             | 723 |
| <i>Naming Conventions for Amazon Redshift</i>                          | 723 |
| <i>Data Types for Amazon Redshift</i>                                  | 724 |
| Overview                                                               | 724 |
| Amazon Redshift Null Values                                            | 724 |
| LIBNAME Statement Data Conversions                                     | 724 |
| <i>Sample Programs for Amazon Redshift</i>                             | 726 |

---

# System Requirements for SAS/ACCESS Interface to Amazon Redshift

You can find information about system requirements for SAS/ACCESS Interface to Amazon Redshift in the following locations:

- [System Requirements for SAS/ACCESS Interface to Amazon Redshift with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to Amazon Redshift

For available SAS/ACCESS features, see “[SAS/ACCESS Interface to Amazon Redshift: Supported Features](#)” on page 97. For more information about Amazon Redshift, see your Amazon Redshift documentation.

---

**Note:** The SAS/ACCESS Interface to Amazon Redshift was implemented in the April 2016 release of SAS/ACCESS.

---

Beginning in [SAS Viya 3.3](#), SAS/ACCESS Interface to Amazon Redshift includes SAS Data Connector to Amazon Redshift. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“Amazon Redshift Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

---

# LIBNAME Statement for the Amazon Redshift Engine

---

## Overview

For general information, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing the Amazon Redshift interface.

**LIBNAME** *libref redshift <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

*redshift*

specifies the SAS/ACCESS engine name for the Amazon Redshift interface.

*connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are defined.

---

**Note:** All of these connection options are valid when used in the CONNECT statement with the SQL procedure.

---

**SERVER=<'>*Amazon Redshift-server-name*<'>**

specifies the server name or IP address of the Amazon Redshift server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**DATABASE=<'>*Amazon Redshift-database-name*<'>**

specifies the name of the database on the Amazon Redshift server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORT=port**

specifies the port number that is used to connect to the specified Amazon Redshift server.

Default: 5439

**USER=<'>Amazon Redshift-user-name<'>**

specifies the Amazon Redshift user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**PASSWORD=<'>Amazon Redshift-password<'>**

specifies the password that is associated with your Amazon Redshift user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: PASS=, PW=, PWD=

**DSN=<'>Amazon Redshift-data-source<'>**

specifies the configured Amazon Redshift ODBC data source to which you want to connect. Use this option if you have existing Amazon Redshift ODBC data sources that are configured on your client. This method requires additional setup—either through the Administrator control panel on Windows platforms or through the odbc.ini file or a similarly named configuration file on UNIX platforms. It is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

**COMPLETE=<'>CLI-connection-string<'>**

specifies connection information for your database for PCs only. Separate multiple options with a semicolon. When a successful connection is made, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, you are prompted with a dialog box that displays the values from the COMPLETE= connection string. You can edit any field before you connect to the database. See your PostgreSQL documentation for more details.

This option is not available on UNIX platforms.

Support for this connection option was added in SAS Viya 3.4.

**PROMPT=<'>PostgreSQL-connection-options<'>**

specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately try to connect to the DBMS. Instead, it displays a dialog box in SAS Display Manager that contains the values that you entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the database.

Restriction: This option is not available on UNIX platforms.

Support for this connection option was added in SAS Viya 3.4.

**NOPROMPT=<'>PostgreSQL-connection-options<'>**

specifies connection options for your database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned. No dialog box is displayed to help you with the connection string.

Support for this connection option was added in SAS Viya 3.4.

*LIBNAME-options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Amazon Redshift with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For more information, see [LIBNAME Options for Relational Databases on page 134](#).

*Table 16.1 SAS/ACCESS LIBNAME Options for Amazon Redshift*

| Option               | Default Value                                           | Valid in CONNECT |
|----------------------|---------------------------------------------------------|------------------|
| ACCESS=              | none                                                    |                  |
| AUTHDOMAIN=          | none                                                    |                  |
| AUTOCOMMIT=          | YES                                                     | •                |
| BL_AWS_CONFIG_FILE=  | ~/.aws/config                                           |                  |
| BL_AWS_PROFILE_NAME= | none                                                    |                  |
| BL_BUCKET=           | none                                                    |                  |
| BL_COMPRESS=         | NO                                                      |                  |
| BL_CONFIG=           | none                                                    |                  |
| BL_DEFAULT_DIR=      | location that is specified by the UTILLOC system option |                  |
| BL_DELETE_DATAFILE=  | YES                                                     |                  |
| BL_DELIMITER=        | the bell character (ASCII 0x07)                         |                  |
| BL_ENCKEY=           | none                                                    |                  |
| BL_KEY=              | none                                                    |                  |
| BL_NUM_READ_THREADS= | 4                                                       |                  |
| BL_OPTIONS=          | none                                                    |                  |
| BL_REGION=           | none                                                    |                  |
| BL_SECRET=           | none                                                    |                  |
| BL_TOKEN=            | none                                                    |                  |

| Option               | Default Value                                                                        | Valid in CONNECT |
|----------------------|--------------------------------------------------------------------------------------|------------------|
| BL_USE_ESCAPE=       | NO                                                                                   |                  |
| BL_USE_SSL=          | YES                                                                                  |                  |
| BULKUNLOAD=          | NO                                                                                   |                  |
| CONNECTION=          | SHAREDREAD                                                                           | •                |
| CONNECTION_GROUP=    | none                                                                                 |                  |
| CONOPTS=             | none                                                                                 | •                |
| CURSOR_TYPE=         | DYNAMIC                                                                              | •                |
| DBCLIENT_MAX_BYTES=  | matches the maximum number of bytes per single character of the SAS session encoding | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows)                                   |                  |
| DBCONINIT=           | none                                                                                 | •                |
| DBCONTERM=           | none                                                                                 |                  |
| DBCREATE_TABLE_OPTS= | none                                                                                 |                  |
| DBGEN_NAME=          | DBMS                                                                                 |                  |
| DBLIBINIT=           | none                                                                                 |                  |
| DBLIBTERM=           | none                                                                                 |                  |
| DBMAX_TEXT=          | 1024                                                                                 | •                |
| DBMSTEMP=            | NO                                                                                   |                  |
| DBNULLKEYS=          | YES                                                                                  |                  |
| DBPROMPT=            | NO                                                                                   |                  |
| DBSASLABEL=          | COMPAT                                                                               |                  |
| DBSERVER_MAX_BYTES=  | none                                                                                 | •                |
| DEFER=               | NO                                                                                   |                  |

| Option                    | Default Value                                                   | Valid in CONNECT |
|---------------------------|-----------------------------------------------------------------|------------------|
| DELETE_MULT_ROWS=         | NO                                                              |                  |
| DIRECT_EXE=               | none                                                            |                  |
| DIRECT_SQL=               | YES                                                             |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                              |                  |
| INSERT_SQL=               | none                                                            |                  |
| INSERTBUFF=               | 250                                                             |                  |
| KEYSET_SIZE=              | 0                                                               | •                |
| LOGIN_TIMEOUT=            | 0                                                               | •                |
| MULTI_DATASRC_OPT=        | NONE                                                            |                  |
| POST_STMT_OPTS=           | none                                                            |                  |
| PRESERVE_COL_NAMES=       | NO                                                              |                  |
| PRESERVE_TAB_NAMES=       | NO                                                              |                  |
| QUALIFIER=                | none                                                            |                  |
| QUERY_TIMEOUT=            | 0                                                               | •                |
| QUOTE_CHAR=               | none                                                            |                  |
| READ_ISOLATION_LEVEL=     | RC (see “Locking in the Amazon Redshift Interface” on page 721) | •                |
| READ_LOCK_TYPE=           | ROW                                                             | •                |
| READBUFF=                 | 0                                                               | •                |
| REREAD_EXPOSURE=          | NO                                                              |                  |
| SCHEMA=                   | none                                                            |                  |
| SPOOL=                    | YES                                                             |                  |
| SQL_FUNCTIONS=            | none                                                            |                  |
| SQL_FUNCTIONS_COPY=       | none                                                            |                  |

| Option                  | Default Value                                                   | Valid in CONNECT |
|-------------------------|-----------------------------------------------------------------|------------------|
| SQLGENERATION=          | none                                                            |                  |
| STRINGDATES=            | NO                                                              | •                |
| SUB_CHAR=               | none                                                            |                  |
| TRACE=                  | NO                                                              | •                |
| TRACEFILE=              | none                                                            | •                |
| UPDATE_ISOLATION_LEVEL= | RC (see “Locking in the Amazon Redshift Interface” on page 721) | •                |
| UPDATE_LOCK_TYPE=       | ROW                                                             |                  |
| UTILCONN_TRANSIENT=     | NO                                                              |                  |

## Best Practice: Specify the Schema

By default, the value of schema is assumed to be PUBLIC in the connection to the database when you omit SCHEMA= from a LIBNAME statement. However, if you omit SCHEMA= from a LIBNAME statement, other SAS procedures, such as the DATASETS procedure, assume that the schema is the same as the user name. This might lead to unexpected results when you call the DATASETS procedure. In addition, to use the KILL functionality with the DATASETS procedure, you must specify SCHEMA= in the LIBNAME statement. Therefore, the best practice is to always specify the value for SCHEMA= in a LIBNAME statement that connects to an Amazon Redshift database.

## Amazon Redshift LIBNAME Statement Examples

In this example, SERVER=, DATABASE=, USER=, and PASSWORD= are connection options. No DSN style is specified. This is the default method, which is recommended.

```
libname A1 redshift server=mysrv1 port=5439
      user=myusr1 password='mypwd1' schema=public database=mydb1;
```

This example requires that you specify a DSN style.

```
libname B1 redshift dsn=rsfttest
      user=myusr1 password=mypwd1 schema=public;
```

# Data Set Options for Amazon Redshift

For more information, see [Data Set Options for Relational Databases on page 370](#).

**Table 16.2** SAS/ACCESS Data Set Options for Amazon Redshift

| Option                   | Default Value                                                            |
|--------------------------|--------------------------------------------------------------------------|
| BL_AWS_CONFIG_FILE=      | ~/.aws/config                                                            |
| BL_AWS_CREDENTIALS_FILE= | ~/.aws/credentials                                                       |
| BL_AWS_PROFILE_NAME=     | none                                                                     |
| BL_BUCKET=               | none                                                                     |
| BL_COMPRESS=             | NO                                                                       |
| BL_CONFIG=               | none                                                                     |
| BL_DEFAULT_DIR=          | temporary file directory that is specified by the UTILLOC= system option |
| BL_DELETE_DATAFILE=      | YES                                                                      |
| BL_DELIMITER=            | comma (',')                                                              |
| BL_ENCKEY=               | LIBNAME option value                                                     |
| BL_IAM_ROLE=             | none                                                                     |
| BL_KEY=                  | none                                                                     |
| BL_NUM_DATAFILES=        | 2                                                                        |
| BL_NUM_READ_THREADS=     | LIBNAME option value                                                     |
| BL_OPTIONS=              | none                                                                     |
| BL_REGION=               | none                                                                     |
| BL_SECRET=               | none                                                                     |
| BL_TOKEN=                | none                                                                     |
| BL_USE_ESCAPE=           | NO                                                                       |

| Option                    | Default Value                                    |
|---------------------------|--------------------------------------------------|
| BL_USE_MANIFEST=          | YES                                              |
| BL_USE_SSL=               | YES                                              |
| BULKLOAD=                 | NO                                               |
| BULKUNLOAD=               | LIBNAME option value                             |
| CURSOR_TYPE=              | LIBNAME option value                             |
| DBCOMMIT=                 | LIBNAME option value                             |
| DBCONDITION=              | none                                             |
| DBCREATE_TABLE_OPTS=      | LIBNAME option value                             |
| DBFORCE=                  | NO                                               |
| DBGEN_NAME=               | DBMS                                             |
| DBLABEL=                  | NO                                               |
| DBLARGETABLE=             | none                                             |
| DBMAX_TEXT=               | 1024                                             |
| DBNULL=                   | YES                                              |
| DBNULLKEYS=               | LIBNAME option value                             |
| DBPROMPT=                 | LIBNAME option value                             |
| DBSASLABEL=               | COMPAT                                           |
| DBSASTYPE=                | See “Data Types for Amazon Redshift” on page 724 |
| DBTYPE=                   | See “Data Types for Amazon Redshift” on page 724 |
| ERRLIMIT=                 | 1                                                |
| IGNORE_READ_ONLY_COLUMNS= | NO                                               |
| INSERT_SQL=               | LIBNAME option value                             |
| INSERTBUFF=               | LIBNAME option value                             |

| Option                  | Default Value        |
|-------------------------|----------------------|
| KEYSET_SIZE=            | LIBNAME option value |
| NULLCHAR=               | SAS                  |
| NULLCHARVAL=            | a blank character    |
| POST_STMT_OPTS=         | none                 |
| POST_TABLE_OPTS=        | none                 |
| PRE_STMT_OPTS=          | none                 |
| PRE_TABLE_OPTS=         | none                 |
| PRESERVE_COL_NAMES=     | LIBNAME option value |
| QUALIFIER=              | LIBNAME option value |
| QUERY_TIMEOUT=          | LIBNAME option value |
| READBUFF=               | LIBNAME option value |
| READ_LOCK_TYPE=         | ROW                  |
| SASDATEFMT=             | none                 |
| SCHEMA=                 | LIBNAME option value |
| SUB_CHAR=               | none                 |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |
| UPDATE_LOCK_TYPE=       | LIBNAME option value |

---

# SQL Pass-Through Facility Specifics for Amazon Redshift

---

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Amazon Redshift interface.

- The *dbms-name* is **redshift**.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Amazon Redshift. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default **redshift** alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).
- Amazon Redshift does not support indexes. Statements that pertain to indexes, such as the CREATE INDEX statement, are not supported.

---

## CONNECT Statement Examples for Amazon Redshift

This example connects to Amazon Redshift and then disconnects from it.

```
proc sql noerrorstop;
    connect to redshift as x1(server=mysrv1 port=5439
        user=mysurl password='mypwd1' database=mydb1);
    disconnect from x1;
    quit;
```

This next example connects to Amazon Redshift, executes some SQL statements, and then disconnects from Amazon Redshift.

```
proc sql noerrorstop;
    connect to redshift as x1(server=mysrv1 port=5439
        user=mysurl password='mypwd1' database=mydb1);

    execute ( CREATE TABLE t1 ( no int primary key, state varchar(10) ) )
    by x1;
    execute ( INSERT INTO t1 values (1, 'USA') ) by x1;
```

```

execute ( INSERT INTO t1 values (2, 'CHN') ) by x1;
select * from connection to x1 (SELECT * FROM t1 ORDER BY no);

disconnect from x1;
quit;

```

# Passing SAS Functions to Amazon Redshift

SAS/ACCESS Interface to Amazon Redshift passes the following SAS functions to Amazon Redshift for processing. Where the Amazon Redshift function name differs from the SAS function name, the Amazon Redshift name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on [page 58](#).

|                          |                    |
|--------------------------|--------------------|
| ABS                      | MINUTE             |
| ARCOS (ACOS)             | MOD (see note)     |
| ARSIN (ASIN)             | MONTH              |
| ATAN                     | QTR                |
| ATAN2                    | REPEAT             |
| BYTE                     | SECOND             |
| CEIL                     | SIGN               |
| COALESCE                 | SIN                |
| COS                      | SQRT               |
| DAY                      | STD (STDDEV_SAMP)  |
| EXP                      | STRIP (BTRIM)      |
| FLOOR                    | SUBSTR (SUBSTRING) |
| HOUR                     | TAN                |
| INDEX (POSITION, STRPOS) | TRANWRD (REPLACE)  |
| LENGTH                   | TRIMN (RTRIM)      |
| LENGTHN (LENGTH)         | UPCASE (UPPER)     |
| LOG (LN)                 | VAR (VAR_SAMP)     |
| LOG10 (LOG)              | WEEKDAY            |
| LOWCASE (LOWER)          | YEAR               |

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on [page 59](#).

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to Amazon Redshift. Because of incompatibility in

date and time functions between Amazon Redshift and SAS, Amazon Redshift might not process them correctly. Check your results to determine whether these functions are working as expected.

|                     |                            |
|---------------------|----------------------------|
| COMPRESS            | LENGTHC (CHARACTER_LENGTH) |
| DATE (CURRENT_DATE) | ROUND                      |
| DATEPART            | TODAY (CURRENT_DATE)       |
| DATETIME (GETDATE)  | TRANSLATE                  |

## Passing Joins to Amazon Redshift

In order for a multiple-libref join to be passed to Amazon Redshift, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- database (DATABASE=)
- port (PORT=)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

## Bulk Loading for Amazon Redshift

### Overview

Bulk loading is the fastest way to insert large numbers of rows into an Amazon Redshift table. To use the bulk-load facility, set the **BULKLOAD=** data set option to YES. You can also perform bulk unloading (data retrieval) from Amazon Redshift. To enable bulk unloading of data, set the **BULKUNLOAD= LIBNAME** option to YES.

The bulk-load facility uses the Amazon Simple Storage Service (S3) tool to move data to and from the client to the Amazon Redshift database. You can find more information about managing data on Amazon Redshift on the [Amazon Web Services web site](#).

---

## Data Set Options with Bulk Loading

Here are the Amazon Redshift bulk-load data set options. For detailed information about these options, see “[Overview](#)” on page 370. For bulk loading, you must use the data set options below.

- [BL\\_AWS\\_CONFIG\\_FILE=](#)
- [BL\\_AWS\\_PROFILE\\_NAME=](#)
- [BL\\_BUCKET=](#)
- [BL\\_COMPRESS=](#)
- [BL\\_CONFIG=](#)
- [BL\\_DEFAULT\\_DIR=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DELIMITER=](#)
- [BL\\_KEY=](#)
- [BL\\_NUM\\_DATAFILES=](#)
- [BL\\_OPTIONS=](#)
- [BL\\_REGION=](#)
- [BL\\_SECRET=](#)
- [BL\\_TOKEN=](#)
- [BL\\_USE\\_MANIFEST=](#)
- [BL\\_USE\\_ESCAPE=](#)
- [BL\\_USE\\_SSL=](#)
- [BULKLOAD=](#)

---

## LIBNAME Options for Bulk Unloading

Here are the LIBNAME options that are available for bulk unloading (data retrieval) from Amazon Redshift. To enable bulk unloading, specify BULKUNLOAD=YES. For bulk unloading, you can use either the LIBNAME options below or the corresponding data set options.

**Note:** Support for these LIBNAME options was added in SAS 9.4M6.

- 
- [BL\\_AWS\\_CONFIG\\_FILE=](#)
  - [BL\\_AWS\\_PROFILE\\_NAME=](#)
  - [BL\\_BUCKET=](#)
  - [BL\\_COMPRESS=](#)

- BL\_CONFIG=
- BL\_DEFAULT\_DIR=
- BL\_DELETE\_DATAFILE=
- BL\_DELIMITER=
- BL\_KEY=
- BL\_OPTIONS=
- BL\_REGION=
- BL\_SECRET=
- BL\_TOKEN=
- BL\_USE\_ESCAPE=
- BL\_USE\_SSL=
- BULKUNLOAD=

## Examples of Bulk Loading

In the following example, a user is creating a new data set, Myclass, in the Amazon Redshift bucket, myBucket. The AWS bucket is stored within the Amazon US East (us-east-1) region cluster. Intermediate bulk-load files are created in the /tmp directory on the user's client machine. Security credentials are provided with the BL\_KEY= and BL\_SECRET= data set options.

The 's' in the SASTRACE option activates output of timing information in the SAS log about the bulk-loading operation.

```
options sastrace=',,ds' sastraceloc=saslog nostsuffix;

libname libred redshift server=rsserver db=rsdb user=myuserID
pwd=myPwd port=5439;

data libred.myclass(
  bulkload=yes
    bl_bucket=myBucket
    bl_key=99999
    bl_secret=12345
    bl_default_dir='/tmp'
    bl_region='us-east-1');
  set sashelp.class;
run;
```

In this example, a user is referencing a subdirectory, school, in the Amazon Redshift bucket, myBucket. As in the previous example, the AWS bucket is stored within the Amazon US East (us-east-1) region cluster. A secure connection to Amazon S3 is established using TLS encryption. The contents of the Class table are read into the Myclass data set.

```
libname mydb redshift server=rsserver db=rsdb user=myuserID pwd=myPwd
port=5439;

data mydb.myclass (bulkload=yes
```

```
bl_bucket='myBucket/school'  
bl_region='us-east-1'  
bl_use_ssl=yes  
) ;  
set sashelp.class;  
run;
```

---

## Encryption with Amazon Redshift

You can use encryption when you perform bulk loading or bulk unloading between Amazon Redshift and SAS. Here is how to do this.

- 1 Assign a user-friendly name to an encryption key by using [PROC S3](#).

---

**Note:** Only IAM:KMS keys are supported for SAS/ACCESS Interface to Amazon Redshift.

---

- 2 Enable bulk loading with the [BULKLOAD= data set option](#), or enable bulk unloading with the [BULKUNLOAD= LIBNAME option](#) or [BULKUNLOAD= data set option](#).
- 3 Specify the [BL\\_ENCKEY= LIBNAME option](#) or [BL\\_ENCKEY= data set option](#) to indicate that data files should be encrypted. The value for BL\_ENCKEY= is the user-friendly name that you assigned to an encryption key by using PROC S3.
- 4 Specify any additional bulk-load options. For more information, see “[Bulk Loading for Amazon Redshift](#)” on page 718.

---

## Locking in the Amazon Redshift Interface

The following LIBNAME and data set options let you control how the Amazon Redshift interface handles locking. For general information, see “[LIBNAME Options for Relational Databases](#)” on page 134.

READ\_LOCK\_TYPE= ROW

UPDATE\_LOCK\_TYPE= ROW

READ\_ISOLATION\_LEVEL= S | RC

The Amazon Redshift ODBC driver manager supports the S and RC isolation levels that are defined in this table.

**Table 16.3** Isolation Levels for Amazon Redshift

| Isolation Level       | Definition                                                                                                                                                                                                                       |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| S (serializable)      | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.                                                                                                                                                               |
| RR (repeatable read)  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.                                                                                                                                                     |
| RC (read committed)   | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.                                                                                                                                                     |
| RU (read uncommitted) | Allows dirty Reads, nonrepeatable Reads, and phantom Reads.                                                                                                                                                                      |
| V (versioning)        | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads. These transactions are serializable, but higher concurrency is possible than with the serializable isolation level. Typically, a nonlocking protocol is used. |

Here are how the terms in the table are defined.

#### *Dirty reads*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it can see changes that are made by those concurrent transactions even before they are committed.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

#### *Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist, a phantom.

`UPDATE_ISOLATION_LEVEL= S | RC`

The Amazon Redshift ODBC driver manager supports the S and RC isolation levels that are defined in the preceding table.

---

# Working with NLS Characters in Amazon Redshift Data

If your data contains NLS characters, you might receive errors in the SAS log that stops processing for a DATA step. If you prefer to continue with processing, you can enable the `SAS_REDSHIFT_UPDATE_WARNING=` environment variable. When you specify `SAS_REDSHIFT_UPDATE_WARNING=1`, then a warning is printed to the SAS log and processing continues.

The warning might resemble the following example:

```
WARNING: During update: [SAS] [ODBC Redshift Wire Protocol driver]No rows were  
affected by UPDATE/DELETE WHERE CURRENT OF cursor  
emulation
```

---

# Naming Conventions for Amazon Redshift

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Amazon Redshift interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The `PRESERVE_TAB_NAMES=` and `PRESERVE_COL_NAMES=` options determine how this interface handles case sensitivity, spaces, and special characters. (For general information, see [LIBNAME Statement for Relational Databases on page 127](#).) Amazon Redshift is not case sensitive, and all names default to lowercase.

Amazon Redshift objects include tables, views, and columns. They follow these naming conventions.

- A name can contain up to 128 characters.
- The first character in a name can be a letter, @, \_, or #.
- A name cannot be an Amazon Redshift reserved word, such as `WHERE` or `VIEW`.

- A name cannot be the same as another Amazon Redshift object that has the same type.

---

# Data Types for Amazon Redshift

---

## Overview

Every column in a table has a name and a data type. The data type tells Amazon Redshift how much physical storage to set aside for the column and the form in which the data is stored.

For more information about Amazon Redshift data types and to determine which data types are available for your version of Amazon Redshift, see your Amazon Redshift documentation.

---

## Amazon Redshift Null Values

Amazon Redshift uses a special value called NULL to identify an absence of information for a column. When SAS/ACCESS reads an Amazon Redshift NULL value, it interprets it as a SAS missing value.

You can specify a column in an Amazon Redshift table so that it requires data. To do this in SQL, you specify a column as NOT NULL. This tells SQL to allow a row to be added to a table only if a value exists for the column. For example, when you assign NOT NULL to the CUSTOMER column in the SASDEMO.CUSTOMER table, you cannot add a row unless there is a value for CUSTOMER.

**TIP** When creating a table with SAS/ACCESS, you can use the DBNULL= data set option to indicate whether NULL is a valid value for specified columns.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how SAS missing character values are handled, use the `NULCHAR=` and `NULLCHARVAL=` data set options.

---

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Amazon Redshift assigns to SAS variables when using the [LIBNAME statement](#) to read from an Amazon Redshift table. These default formats are based on Amazon Redshift

column attributes. SAS/ACCESS does not support Amazon Redshift data types that do not appear in this table.

**Table 16.4** Default SAS Formats for Amazon Redshift Data Types

| Amazon Redshift Data Type        | ODBC Data Type           | Default SAS Format                                                                      |
|----------------------------------|--------------------------|-----------------------------------------------------------------------------------------|
| CHAR( <i>n</i> ) <sup>1</sup>    | SQL_CHAR                 | \$ <i>w</i> .                                                                           |
| VARCHAR( <i>n</i> ) <sup>1</sup> | SQL_LONGCHAR( <i>n</i> ) |                                                                                         |
| DECIMAL                          | SQL_DECIMAL              | <i>w</i> or <i>w.d</i> or none if you do not specify <i>w</i> and <i>d</i> <sup>2</sup> |
| INTEGER                          | SQL_INTEGER              | 11.                                                                                     |
| SMALLINT                         | SQL_SMALLINT             | 6.                                                                                      |
|                                  | SQL_TINYINT              | 4.                                                                                      |
| REAL                             | SQL_REAL                 | none                                                                                    |
| FLOAT                            | SQL_FLOAT                |                                                                                         |
| DOUBLE PRECISION                 | SQL_DOUBLE               |                                                                                         |
| BIGINT                           | SQL_BIGINT               | 20.                                                                                     |
| BOOLEAN                          | SQL_BOOLEAN              | 1.                                                                                      |
| DATE                             | SQL_TYPE_DATE            | DATE9.                                                                                  |
| TIMESTAMP                        | SQL_TYPE_TIMESTAMP       | DATETIME formats                                                                        |

<sup>1</sup> *n* in Amazon Redshift character data types is equivalent to *w* in SAS formats.

<sup>2</sup> *m* and *n* in Amazon Redshift numeric data types are equivalent to *w* and *d* in SAS formats.

The following table shows the default data types that SAS/ACCESS Interface to Amazon Redshift uses when creating tables. The Amazon Redshift engine lets you specify nondefault data types by using the DBTYPE= data set option.

**Table 16.5** Default SAS Formats for Amazon Redshift Data Types When Creating Tables in Amazon Redshift

| Default SAS Format | ODBC Data Type                                                         | Amazon Redshift Data Type          |
|--------------------|------------------------------------------------------------------------|------------------------------------|
| <i>w.d</i>         | SQL_DOUBLE or<br>SQL_NUMERIC using <i>m,n</i><br>if the DBMS allows it | NUMERIC( <i>m,n</i> ) <sup>2</sup> |

| Default SAS Format | ODBC Data Type             | Amazon Redshift Data Type        |
|--------------------|----------------------------|----------------------------------|
| \$w.               | SQL_VARCHAR using <i>n</i> | VARCHAR( <i>n</i> ) <sup>1</sup> |
| DATETIME formats   | SQL_TIMESTAMP              | TIMESTAMP <sup>3</sup>           |
| DATE formats       | SQL_DATE                   | DATE                             |
| TIME formats       | SQL_TIMESTAMP              | TIMESTAMP <sup>4</sup>           |

<sup>1</sup> *n* in Amazon Redshift character data types is equivalent to *w* in SAS formats.

<sup>2</sup> *m* and *n* in Amazon Redshift numeric data types are equivalent to *w* and *d* in SAS formats.

<sup>3</sup> Although the Amazon Redshift engine supports TIMESTAMP, it has no TIMESTAMP WITH TIMEZONE data type that maps to the corresponding Amazon Redshift data type.

<sup>4</sup> A SAS time value is converted to a TIMESTAMP value of 1960-01-01 <time> in Amazon Redshift.

## Sample Programs for Amazon Redshift

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to Aster

---

|                                                                    |            |
|--------------------------------------------------------------------|------------|
| <b>System Requirements for SAS/ACCESS Interface to Aster .....</b> | <b>728</b> |
| <b>Introduction to SAS/ACCESS Interface to Aster .....</b>         | <b>728</b> |
| <b>LIBNAME Statement for the Aster Engine .....</b>                | <b>729</b> |
| Overview .....                                                     | 729        |
| Arguments .....                                                    | 729        |
| Aster LIBNAME Statement Examples .....                             | 733        |
| <b>Data Set Options for Aster .....</b>                            | <b>734</b> |
| <b>SQL Pass-Through Facility Specifics for Aster .....</b>         | <b>736</b> |
| Key Information .....                                              | 736        |
| CONNECT Statement Example .....                                    | 737        |
| Special Catalog Queries .....                                      | 737        |
| <b>Autopartitioning Scheme for Aster .....</b>                     | <b>738</b> |
| Overview .....                                                     | 738        |
| Autopartitioning Restrictions .....                                | 738        |
| Nullable Columns .....                                             | 739        |
| Using WHERE Clauses .....                                          | 739        |
| Using DBSLICEPARM= .....                                           | 739        |
| Using DBSLICE= .....                                               | 739        |
| <b>Temporary Table Support for Aster .....</b>                     | <b>740</b> |
| <b>Passing SAS Functions to Aster .....</b>                        | <b>740</b> |
| <b>Passing Joins to Aster .....</b>                                | <b>741</b> |
| <b>Bulk Loading and Unloading for Aster .....</b>                  | <b>742</b> |
| Loading .....                                                      | 742        |
| Data Set Options with Bulk Loading .....                           | 742        |
| Examples .....                                                     | 742        |
| Unloading .....                                                    | 743        |
| Example .....                                                      | 743        |
| <b>Naming Conventions for Aster .....</b>                          | <b>744</b> |
| <b>Data Types for Aster .....</b>                                  | <b>745</b> |
| Overview .....                                                     | 745        |
| Supported Aster Data Types .....                                   | 745        |
| LIBNAME Statement Data Conversions .....                           | 746        |

# System Requirements for SAS/ACCESS Interface to Aster

You can find information about system requirements for SAS/ACCESS Interface to Aster in the following locations:

- [System Requirements for SAS/ACCESS Interface to Aster with SAS 9.4](#)
- [Third-Party Software Requirements for Use with SAS 9.4](#)

## Introduction to SAS/ACCESS Interface to Aster

For available SAS/ACCESS features, see [Aster supported features on page 98](#). For more information about Aster, see your Aster documentation.

---

**Note:** SAS/ACCESS Interface to Aster is not included with SAS Viya.

---

**Note:** Beginning with [SAS 9.4M8](#), SAS/ACCESS Interface to Aster is no longer available. If you have an existing instance of SAS/ACCESS Interface to Aster and plan to upgrade to SAS 9.4M8 or later, SAS recommends that you first unconfigure and uninstall it. For more information, see [Unconfiguring and Uninstalling Retired Products](#). Documentation remains for existing customers who have not upgraded to SAS 9.4M8.

---

---

# LIBNAME Statement for the Aster Engine

---

## Overview

This section describes the LIBNAME statement options that SAS/ACCESS Interface to Aster supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Aster.

**LIBNAME** *libref aster <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *aster*

specifies the SAS/ACCESS engine name for the Aster interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the Aster database in several ways. Specify only one of these methods for each connection because they are mutually exclusive.

- SERVER=, DATABASE=, PORT=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD=
- NOPROMPT=
- PROMPT=
- REQUIRED=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**SERVER=<'>server-name<'>**  
 specifies the host name or IP address where the Aster database is running. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**DATABASE=<'>database-name<'>**  
 Alias: DB=  
 specifies the Aster database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**PORT=port**  
 specifies the port number that is used to connect to the specified Aster database. If you do not specify a port, the default port 2406 is used.

**USER=<'>Aster-user-name<'>**  
 specifies the Aster user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**PASSWORD=<'>Aster-password<'>**  
 Alias: PASS=, PW=, PWD=  
 specifies the password that is associated with your Aster user ID. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**DSN=<'>Aster-data-source<'>**  
 specifies the configured Aster ODBC data source to which you want to connect. Use this option if you have existing Aster ODBC data sources that are configured on your client. This method requires additional setup, either through the ODBC Administrator control panel on Windows platforms or through the odbc.ini file on UNIX platforms. So it is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

Requirement: If you use this method, you just specify the user ID and password in the LIBNAME statement, even if these are already specified in the ODBC data source.

**NOPROMPT=<'>Aster-ODBC-connection-options<'>**  
 specifies connection options for your database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned. No dialog box is displayed to help you with the connection string.

**PROMPT=<'> Aster-ODBC-connection-options<'>**  
 specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately try to connect to the DBMS. It instead displays a dialog box in SAS Display Manager that contains the values that you entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the database.

Restriction: This option is not supported on UNIX platforms.

**REQUIRED=<'>Aster-ODBC-connection-options<'>**  
 specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough

correct connection options, a dialog box prompts you for the connection options. REQUIRED= lets you modify only required fields in the dialog box. This option is not supported on UNIX platforms.

***LIBNAME -options***

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Aster with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 17.1** SAS/ACCESS LIBNAME Options for Aster

| Option              | Default Value                                                                                          | Valid in<br>CONNECT |
|---------------------|--------------------------------------------------------------------------------------------------------|---------------------|
| ACCESS=             | none                                                                                                   |                     |
| AUTHDOMAIN=         | none                                                                                                   |                     |
| AUTOCOMMIT=         | operation-specific                                                                                     | •                   |
| BL_DEFAULT_DIR=     | temporary file<br>directory that is<br>specified by the<br>UTILLOC= system option                      |                     |
| BULKUNLOAD=         | NO                                                                                                     |                     |
| CONNECTION=         | UNIQUE                                                                                                 | •                   |
| CONNECTION_GROUP=   | none                                                                                                   | •                   |
| CONOPTS=            | none                                                                                                   | •                   |
| DBCLIENT_MAX_BYTES= | matches the<br>maximum<br>number of bytes<br>per single<br>character of the<br>SAS session<br>encoding | •                   |
| DBCOMMIT=           | 1000 (when<br>inserting rows), 0<br>(when updating<br>rows)                                            |                     |
| DBCONINIT=          | none                                                                                                   | •                   |
| DBCONTERM=          | none                                                                                                   | •                   |

| Option                    | Default Value                                | Valid in CONNECT |
|---------------------------|----------------------------------------------|------------------|
| DBCREATE_TABLE_OPTS=      | none                                         |                  |
| DBGEN_NAME=               | DBMS                                         | •                |
| DBINDEX=                  | NO                                           |                  |
| DBLIBINIT=                | none                                         |                  |
| DBLIBTERM=                | none                                         |                  |
| DBMAX_TEXT=               | 1024                                         | •                |
| DBMSTEMP=                 | NO                                           |                  |
| DBNULLKEYS=               | YES                                          |                  |
| DBPROMPT=                 | NO                                           | •                |
| DBSASLABEL=               | COMPAT                                       |                  |
| DEFER=                    | NO                                           | •                |
| DELETE_MULT_ROWS=         | NO                                           |                  |
| DIMENSION=                | NO                                           |                  |
| DIRECT_EXE=               | none                                         |                  |
| DIRECT_SQL=               | YES                                          |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                           |                  |
| INSERTBUFF=               | automatically calculated based on row length |                  |
| LOGIN_TIMEOUT=            | 0                                            | •                |
| MULTI_DATASRC_OPT=        | NONE                                         |                  |
| PARTITION_KEY=            | none                                         |                  |
| POST_STMT_OPTS=           | none                                         |                  |
| PRESERVE_COL_NAMES=       | see “Naming Conventions for Aster”           |                  |

| Option              | Default Value                                | Valid in CONNECT |
|---------------------|----------------------------------------------|------------------|
| PRESERVE_TAB_NAMES= | see “Naming Conventions for Aster”           |                  |
| QUERY_TIMEOUT=      | 0                                            | •                |
| QUOTE_CHAR=         | none                                         |                  |
| READBUFF=           | automatically calculated based on row length | •                |
| REREAD_EXPOSURE=    | NO                                           | •                |
| SCHEMA=             | none                                         |                  |
| SPOOL=              | YES                                          |                  |
| SQL_FUNCTIONS=      | none                                         |                  |
| SQL_FUNCTIONS_COPY= | none                                         |                  |
| SQLGENERATION=      | none                                         |                  |
| STRINGDATES=        | NO                                           | •                |
| TRACE=              | NO                                           | •                |
| TRACEFILE=          | none                                         | •                |
| UPDATE_MULT_ROWS=   | NO                                           |                  |
| USE_ODBC_CL=        | NO                                           |                  |
| UTILCONN_TRANSIENT= | NO                                           |                  |

Use the DBCLIENT\_MAX\_BYTES= LIBNAME option only when necessary. When DBCLIENT\_MAX\_BYTES= is not specified in the LIBNAME statement, the table column CHAR length is the same as the SAS CHAR length.

## Aster LIBNAME Statement Examples

In this first example, SERVER=, DATABASE=, USER=, and PASSWORD= are the connection options.

```
LIBNAME mydblib ASTER SERVER=mysrv1 DATABASE=test
```

```

USER=myusr1 PASSWORD=mypwd1;

PROC Print DATA=mydblib.customers;
  WHERE state='CA';
run;

```

In this next example, the DSN= option, the USER= option, and the PASSWORD= option are connection options. The Aster data source is configured in the ODBC Administrator Control Panel on Windows platforms. It is also configured in the odbc.ini file or a similarly named configuration file on UNIX platforms.

```

LIBNAME mydblib aster dsn=aster user=myusr1 password=mypwd1;

PROC Print DATA=mydblib.customers;
  WHERE state='CA';
run;

```

Here is how you can use the NOPROMPT= option.

```

libname x aster NOPROMPT="dsn=aster;";
libname x aster NOPROMPT="DRIVER=aster; server=192.168.28.100;
  uid=username; pwd=password; database=asterdb";

```

This example uses the PROMPT= option. Blanks are also passed down as part of the connection options. Therefore, the specified value must immediately follow the semicolon.

```
libname x aster PROMPT="DRIVER=aster;";
```

The REQUIRED= option is used in this example. If you enter all needed connection options, REQUIRED= does not prompt you for any input.

```
libname x aster REQUIRED="DRIVER=aster; server=192.168.28.100;
  uid=username;pwd=password; database=asterdb ;";
```

As shown above, when `asterdb` (contains a trailing blank) is specified as the database instead of `asterdb` (no trailing blank), this error results:

```

ERROR: CLI error trying to establish connection:
ERROR: Database asterdb does not exist.

```

## Data Set Options for Aster

All SAS/ACCESS data set options in this table are supported for Aster. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

*Table 17.2 SAS/ACCESS Data Set Options for Aster*

| Option                         | Default Value |
|--------------------------------|---------------|
| <code>BL_DATAFILE=</code>      | none          |
| <code>BL_DATAFILE_PATH=</code> | none          |

| <b>Option</b>        | <b>Default Value</b>                         |
|----------------------|----------------------------------------------|
| BL_DBNAME=           | none                                         |
| BL_DELETE_DATAFILE=  | YES                                          |
| BL_DELIMITER=        | \t (the tab symbol)                          |
| BL_ESCAPE=           | \                                            |
| BL_HOST=             | none                                         |
| BL_OPTIONS=          | none                                         |
| BL_PATH=             | none                                         |
| BL_QUOTE=            | "                                            |
| BULKLOAD=            | NO                                           |
| BULKUNLOAD=          | LIBNAME option value                         |
| DBCOMMIT=            | LIBNAME option value                         |
| DBCONDITION=         | none                                         |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                         |
| DBFORCE=             | NO                                           |
| DBGEN_NAME=          | DBMS                                         |
| DBINDEX=             | NO                                           |
| DBKEY=               | none                                         |
| DBLABEL=             | NO                                           |
| DBLARGETABLE=        | none                                         |
| DBMAX_TEXT=          | 1024                                         |
| DBNULL=              | YES                                          |
| DBNULLKEYS=          | LIBNAME option value                         |
| DBPROMPT=            | LIBNAME option value                         |
| DBSASTYPE=           | see “ <a href="#">Data Types for Aster</a> ” |

| Option                    | Default Value              |
|---------------------------|----------------------------|
| DBTYPE=                   | see “Data Types for Aster” |
| DIMENSION=                | NO                         |
| ERRLIMIT=                 | 1                          |
| IGNORE_READ_ONLY_COLUMNS= | NO                         |
| INSERTBUFF=               | LIBNAME option value       |
| NULCHAR=                  | SAS                        |
| NULCHARVAL=               | A blank character          |
| PARTITION_KEY=            | none                       |
| PRESERVE_COL_NAMES=       | LIBNAME option value       |
| QUERY_TIMEOUT=            | LIBNAME option value       |
| READBUFF=                 | LIBNAME option value       |
| SASDATEFMT=               | none                       |
| SCHEMA=                   | LIBNAME option value       |

---

## SQL Pass-Through Facility Specifics for Aster

---

### Key Information

For general information about this feature, see “SQL Pass-Through Facility” on page 690.

Here are the SQL pass-through facility specifics for the Aster interface.

- The *dbms-name* is ASTER.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Aster. If you use multiple simultaneous connections, you must use the *alias* argument to identify the

different connections. If you do not specify an alias, the default ASTER alias is used.

- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME connection options.

## CONNECT Statement Example

This example uses the DBCON alias to connect to `mysrv1` the Aster database and execute a query. The connection alias is optional.

```
proc sql;
  connect to aster as dbcon
    (server=mysrv1 database=test user=myusr1 password=mypwd1);
  select * from connection to dbcon
    (select * from customers WHERE customer like '1%');
quit;
```

## Special Catalog Queries

SAS/ACCESS Interface to Aster supports the following special queries. You can use the queries to call the ODBC-style catalog function application programming interfaces (APIs). Here is the general format of the special queries:

`Aster::SQLAPI'parameter-1', 'parameter-n'`

`Aster::`

is required to distinguish special queries from regular queries. Aster:: is not case sensitive.

`SQLAPI`

is the specific API that is being called. SQLAPI is not case sensitive.

`'parameter n'`

is a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, this call to SQL Tables usually matches table names such as myatest and my\_test:

```
select * from connection to aster (ASTER::SQLTables
  "test", "", "my_test");
```

Use the escape character to search only for the my\_test table.

```
select * from connection to aster (ASTER::SQLTables "test", "", "my\_test");
```

SAS/ACCESS Interface to Aster supports these special queries.

`ASTER::SQLTables <'Catalog', 'Schema', 'Table-name', 'Type'>`

returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.

ASTER::SQLColumns <'Catalog', 'Schema', 'Table-name', 'Column-name'>  
 returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.

ASTER::SQLColumns <'Catalog', 'Schema', 'Table-name', 'Column-name'>  
 returns a list of all columns that match the specified arguments. If you do not specify any argument, all accessible column names and information are returned.

ASTER::SQLPrimaryKeys <'Catalog', 'Schema', 'Table-name' 'Type'>  
 returns a list of all columns that compose the primary key that matches the specified table. A primary key can be composed of one or more columns. If you do not specify any table name, this special query fails.

ASTER::SQLStatistics <'Catalog', 'Schema', 'Table-name'>  
 returns a list of the statistics for the specified table name. You specify options SQL\_INDEX\_ALL and SQL\_ENSURE in the SQLStatistics API call. If you do not specify a table name argument, this special query fails.

ASTER::SQLGetTypeInfo  
 returns information about the data types that the Aster database supports.

ASTER::SQLTablePrivileges<'Catalog', 'Schema', 'Table-name'>  
 returns a list of all tables and associated privileges that match the specified arguments. If no arguments are specified, all accessible table names and associated privileges are returned.

## Autopartitioning Scheme for Aster

### Overview

Autopartitioning for SAS/ACCESS Interface to Aster is a modulo (MOD) function method. For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

### Autopartitioning Restrictions

SAS/ACCESS Interface to Aster places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- SQL\_INTEGER, SQL\_BIT, SQL\_SMALLINT, and SQL\_TINYINT columns are given preference.
- You can use SQL\_DECIMAL, SQL\_DOUBLE, SQL\_FLOAT, SQL\_NUMERIC, and SQL\_REAL columns for partitioning under these conditions:
  - Aster supports converting these types to SQL\_INTEGER by using the INTEGER cast function.

- The precision minus the scale of the column is greater than 0 but less than 10; namely,  $0 < (\text{precision}-\text{scale}) < 10$ .

## Nullable Columns

If you select a nullable column for autopartitioning, the `OR<column-name>IS NULL` SQL statement is appended at the end of the SQL code that is generated for the threaded Read. This ensures that any possible NULL values are returned in the result set. Also, if the column to be used for the partitioning is `SQL_BIT`, the number of threads are automatically changed to two, regardless of the `DBSLICEPARM=` option value.

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in a WHERE clause. For example, this DATA step could not use a threaded Read to retrieve the data. All numeric columns in the table are in the WHERE clause:

```
DATA work.locemp;
  SET trlib.MYEMPS;
  WHERE EMPNUM<=30 and ISTENURE=0 and
        SALARY<=35000 and NUMCLASS>2;
run;
```

## Using DBSLICEPARM=

SAS/ACCESS Interface to Aster defaults to three threads when you use autopartitioning but do not specify a maximum number of threads in to use for the threaded Read. See [DBSLICEPARM= LIBNAME option](#).

## Using DBSLICE=

You might achieve the best possible performance when using threaded Reads by specifying the `DBSLICE= data set option` for Aster in your SAS operation. Using `DBSLICE=` allows connections to individual partitions so that you can configure an Aster data source for each partition. Use this option to specify both the database and the WHERE clause for each partition.

```
proc print data=trilb.MYEMPS (DBSLICE=(DSN1='EMPNUM BETWEEN 1 AND 33'
DSN2='EMPNUM BETWEEN 34 AND 66'
DSN3='EMPNUM BETWEEN 67 AND 100')) ;
run;
```

Using the `DATASOURCE=` option is not required to use `DBSLICE=` option with threaded Reads.

Using DBSLICE= works well when the table that you want to read is not stored in multiple partitions. It gives you flexibility in column selection. For example, if you know that the STATE column in your employee table contains only a few distinct values, you can customize your DBSLICE= option accordingly.

```
data work.locemp;
  set trlib2.MYEMP (DBSLICE= ("STATE='FL'" "STATE='GA'"
    "STATE='SC'" "STATE='VA'" "STATE='NC'"));
  where EMPNUM<=30 and ISTENURE=0 and SALARY<=35000 and NUMCLASS>2;
run;
```

## Temporary Table Support for Aster

SAS/ACCESS Interface to Aster supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## Passing SAS Functions to Aster

SAS/ACCESS Interface to Aster passes the following SAS functions to Aster for processing. Where the Aster function name differs from the SAS function name, the Aster name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                  |                    |
|------------------|--------------------|
| ABS              | MAX                |
| ARCOS (ACOS)     | MIN                |
| ARSIN (ASIN)     | MINUTE (date_part) |
| ATAN             | MLOWCASE (lower)   |
| ATAN2            | MOD (see note)     |
| AVG              | MONTH (date_part)  |
| BYTE (chr)       | QTR (date_part)    |
| CEIL (ceiling)   | REPEAT             |
| COALESCE         | SIGN               |
| COS              | SIN                |
| COT              | SQRT               |
| COUNT            | STRIP (btrim)      |
| DAY (date_part)  | SUBSTR (substring) |
| EXP              | SUM                |
| FLOOR            | TAN                |
| HOUR (date_part) | TRANWRD (replace)  |
| INDEX (strpos)   | TRIMN (rtrim)      |
| LOG (ln)         | UPCASE (upper)     |
| LOG10 (log)      | YEAR (date_part)   |

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to Aster. Due to incompatibility in date and time functions between Aster and SAS, Aster might not process them correctly. Check your results to determine whether these functions are working as expected. For more information, see “[SQL\\_FUNCTIONS= LIBNAME Statement Option](#)” on page 324.

|                    |                   |
|--------------------|-------------------|
| COMPRESS (replace) | ROUND             |
| DATE (now::date)   | TIME (now::time)  |
| DATEPART (cast)    | TIMEPART (cast)   |
| DATETIME (now)     | TODAY (now::date) |
| LENGTH             | TRANSLATE         |

---

## Passing Joins to Aster

For a multiple libref join to pass to Aster, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- database (DATABASE=)
- port (PORT=)
- data source (DSN=, if specified)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

# Bulk Loading and Unloading for Aster

## Loading

Bulk loading is the fastest way to insert large numbers of rows into an Aster table. You must specify `BULKLOAD=YES` to use the bulk-load facility. The bulk-load facility uses the Aster loader client application to move data from the client to the Aster database.

## Data Set Options with Bulk Loading

Here are the Aster bulk-load data set options. For detailed information about these options, see “[About the Data Set Options for Relational Databases](#)” on page 370.

- `BL_DATAFILE=`
- `BL_DATAFILE_PATH=`
- `BL_DBNAME=`
- `BL_DELETE_DATAFILE=`
- `BL_DELIMITER=`
- `BL_HOST=`
- `BL_OPTIONS=`
- `BL_PATH=`
- `BULKLOAD=`
- `BULKUNLOAD=`

## Examples

This example shows how you can use a SAS data set, `SASFILT.FLT98`, to create and load a large Aster table, `FLIGHTS98`.

```

LIBNAME sasflt 'SAS-library';
LIBNAME net_air ASTER user=myusr1 pwd=mypwd1
                     server=air2 database=flights dimension=yes;

PROC sql;
create table net_air.flights98
(bulkload=YES bl_host='queen' bl_path='/home/aster_loader/'
 bl_dbname='beehive')

```

```
as select * from sasflt.flt98;
quit;
```

You can use [BL\\_OPTIONS=](#) to pass specific Aster options to the bulk-loading process.

You can create the same table using a DATA step.

```
data net_air.flights98(bulkload=YES bl_host='queen'
   bl_path='/home/aster_loader/'
   bl_dbname='beehive');
   set sasflt.flt98;
run;
```

You can then append the SAS data set, SASFLT.FLT98, to the existing Aster table, ALLFLIGHTS. SAS/ACCESS Interface to Aster to write data to a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, [BL\\_DELETE\\_DATAFILE=NO](#) causes the engine to retain it after the load completes.

```
PROC append base=net_air.allflights
  (BULKLOAD=YES
   BL_DATAFILE='/tmp/fltdata.dat'
   BL_HOST='queen'
   BL_PATH='/home/aster_loader/'
   BL_DBNAME='beehive'
   BL_DELETE_DATAFILE=NO )
data=sasflt.flt98;
run;
```

## Unloading

Bulk unloading is the fastest way to insert large numbers of rows from an Aster table. To use the bulk-unloading facility, specify BULKUNLOAD=YES. (See [BULKUNLOAD= on page 493](#).) The bulk-unloading facility uses the Aster Remote External Table interface to move data from the client to the Aster Performance Server into SAS.

Here are the Aster bulk-unloading data set options:

- [BL\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DELIMITER=](#)
- [BL\\_USE\\_PIPE=](#)
- [BULKLOAD=](#)
- [BULKUNLOAD=](#)

## Example

This example shows you how you can read a large Aster table to create and populate a SAS data set:

```

libname db aster server ='redqueen.unx.sas.com'
      db='accesstesting' user='dbitest' password='dbigrp1' dimension=yes;

proc sql;
select * from db.employees(bulkunload=YES) ;
create table work.employees as select * from db.employees(bulkunload=YES
      bl_options=' ' BL_DATAFILE='c:\bl_data.dat' BL_USE_PIPE=NO
      BL_DELETE_DATAFILE=NO);
quit;

proc append base=work.employees data=db.empployees(bulkunload=YES
      bl_options=' ' BL_DATAFILE='c:\bl_data.dat' BL_USE_PIPE=NO
      BL_DELETE_DATAFILE=NO);
run;

```

---

## Naming Conventions for Aster

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Aster interface supports table names and column names that contain up to 32 characters. If column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical column names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a name longer than 32 characters. If you have a table name that is greater than 32 characters, it is recommended that you create a table view.

The `PRESERVE_TAB_NAMES=` and `PRESERVE_COL_NAMES=` options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Aster is not case sensitive, so all names default to lowercase.

Aster objects include tables, views, and columns. They follow these conventions.

- A name must be from 1 to 64 characters long.
- A name must begin with a letter (A through Z), diacritic marks, non-Latin characters (200–377 octal) or an underscore (\_).
- To enable case sensitivity, enclose names in quotation marks. All references to quoted names must be enclosed in quotation marks and preserve case sensitivity.
- A name cannot begin with a \_bee prefix. Leading \_bee prefixes are reserved for system objects.
- A name cannot be a reserved word in Aster such as WHERE or VIEW.
- A name cannot be the same as another Aster object that has the same type.

---

# Data Types for Aster

---

## Overview

Every column in a table has a name and a data type. The data type tells Aster how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Aster data types and data conversions.

SAS/ACCESS Interface to Aster does not directly support TIMETZ or INTERVAL types. Any columns using these types are read into SAS as character strings.

For information about Aster data types and to determine which data types are available for your version of Aster, see your Aster documentation.

---

## Supported Aster Data Types

Here are the data types that the Aster engine supports.

- Character data:

CHAR(*n*)    VARCHAR(*n*)

- Numeric data:

BIGINT    DOUBLE | DOUBLE PRECISION

SMALLINT    REAL

INTEGER    DECIMAL | DEC | NUMERIC | NUM

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

---

- Date, time, and timestamp data:

DATE    TIMESTAMP

TIME

**Note:** Be aware that columns of these data types can contain data values that are out of range for SAS.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Aster assigns to SAS variables when using the [LIBNAME statement](#) to read from an Aster table. These default formats are based on Aster column attributes.

**Table 17.3 LIBNAME Statement: Default SAS Formats for Aster Data Types**

| Aster Data Type                  | SAS Data Type | Default SAS Format |
|----------------------------------|---------------|--------------------|
| CHAR( <i>n</i> ) <sup>1</sup>    | character     | \$ <i>w</i> .      |
| VARCHAR( <i>n</i> ) <sup>1</sup> | character     | \$ <i>w</i> .      |
| INTEGER                          | numeric       | 11.                |
| SMALLINT                         | numeric       | 6.                 |
| BIGINT                           | numeric       | 20.                |
| DECIMAL( <i>p,s</i> )            | numeric       | <i>w.d</i>         |
| NUMERIC( <i>p,s</i> )            | numeric       | <i>w.d</i>         |
| REAL                             | numeric       | none               |
| DOUBLE                           | numeric       | none               |
| TIME                             | numeric       | TIME8.             |
| DATE                             | numeric       | DATE9.             |
| TIMESTAMP                        | numeric       | DATETIME25.6       |

<sup>1</sup> *n* in Aster data types is equivalent to *w* in SAS formats.

This table shows the default Aster data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 17.4 LIBNAME Statement: Default Aster Data Types for SAS Variable Formats**

| SAS Variable Format | Aster Data Type       |
|---------------------|-----------------------|
| <i>w.d</i>          | DECIMAL( <i>p,s</i> ) |

| SAS Variable Format | Aster Data Type                  |
|---------------------|----------------------------------|
| other numerics      | DOUBLE                           |
| \$w.                | VARCHAR( <i>n</i> ) <sup>1</sup> |
| datetime formats    | TIMESTAMP                        |
| date formats        | DATE                             |
| time formats        | TIME                             |

<sup>1</sup> *n* in Aster data types is equivalent to *w* in SAS formats.

---

## Sample Programs for Aster

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to DB2 under UNIX and PC Hosts

---

|                                                                                          |     |
|------------------------------------------------------------------------------------------|-----|
| <i>System Requirements for SAS/ACCESS Interface to DB2 under UNIX and PC Hosts</i> ..... | 750 |
| <i>Introduction to SAS/ACCESS Interface to DB2 under UNIX and PC Hosts</i> .....         | 750 |
| <i>LIBNAME Statement for the DB2 Engine under UNIX and PC Hosts</i> .....                | 751 |
| Overview .....                                                                           | 751 |
| Arguments .....                                                                          | 751 |
| DB2 under UNIX and PC Hosts LIBNAME Statement Example .....                              | 756 |
| <i>Data Set Options for DB2 under UNIX and PC Hosts</i> .....                            | 756 |
| <i>SQL Pass-Through Facility Specifics for DB2 under UNIX and PC Hosts</i> .....         | 759 |
| Key Information .....                                                                    | 759 |
| Examples .....                                                                           | 760 |
| Special Catalog Queries .....                                                            | 760 |
| <i>Autopartitioning Scheme for DB2 under UNIX and PC Hosts</i> .....                     | 761 |
| Overview .....                                                                           | 761 |
| Autopartitioning Restrictions .....                                                      | 761 |
| Nullable Columns .....                                                                   | 762 |
| Using WHERE Clauses .....                                                                | 762 |
| Using DBSLICEPARM= .....                                                                 | 762 |
| Using DBSLICE= .....                                                                     | 762 |
| Configuring DB2 EEE Nodes on Physically Partitioned Databases .....                      | 764 |
| <i>Temporary Table Support for DB2 under UNIX and PC Hosts</i> .....                     | 764 |
| <i>Calling Stored Procedures in DB2 under UNIX and PC Hosts</i> .....                    | 765 |
| Overview .....                                                                           | 765 |
| Examples .....                                                                           | 765 |
| <i>DBLOAD Procedure Specifics for DB2 under UNIX and PC Hosts</i> .....                  | 768 |
| Key Information .....                                                                    | 768 |
| Examples .....                                                                           | 769 |
| <i>Passing SAS Functions to DB2 under UNIX and PC Hosts</i> .....                        | 770 |
| <i>Passing Joins to DB2 under UNIX and PC Hosts</i> .....                                | 771 |

|                                                             |     |
|-------------------------------------------------------------|-----|
| <i>Sorting Data That Contains NULL Values</i>               | 772 |
| <i>Bulk Loading for DB2 under UNIX and PC Hosts</i>         | 772 |
| Overview                                                    | 772 |
| Using the LOAD Method                                       | 772 |
| Data Set Options with Bulk Loading                          | 773 |
| Using the IMPORT Method                                     | 773 |
| Using the CLI LOAD Method                                   | 774 |
| Capturing Bulk-Load Statistics into Macro Variables         | 774 |
| Maximizing Load Performance for DB2 under UNIX and PC Hosts | 775 |
| Examples                                                    | 775 |
| <i>Locking in the DB2 under UNIX and PC Hosts Interface</i> | 776 |
| <i>Naming Conventions for DB2 under UNIX and PC Hosts</i>   | 777 |
| <i>Data Types for DB2 under UNIX and PC Hosts</i>           | 778 |
| Overview                                                    | 778 |
| Supported Data Types for DB2 under UNIX and PC Hosts        | 779 |
| Handling Increased Precision with TIMESTAMP Values          | 779 |
| DB2 Null and Default Values                                 | 780 |
| LIBNAME Statement Data Conversions                          | 780 |
| DBLOAD Procedure Data Conversions                           | 782 |
| <i>Sample Programs for DB2 under UNIX and PC Hosts</i>      | 783 |

## System Requirements for SAS/ACCESS Interface to DB2 under UNIX and PC Hosts

You can find information about system requirements for SAS/ACCESS Interface to DB2 under UNIX and PC Hosts in the following locations:

- [System Requirements for SAS/ACCESS Interface to DB2 with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

## Introduction to SAS/ACCESS Interface to DB2 under UNIX and PC Hosts

For available SAS/ACCESS features, see [DB2 under UNIX and PC Hosts supported features on page 99](#). For more information about DB2 under UNIX and PC Hosts, see your DB2 under UNIX and PC Hosts documentation.

Beginning in [SAS Viya 3.3](#), SAS/ACCESS Interface to DB2 under UNIX and PC Hosts includes SAS Data Connector to DB2 under UNIX and PC Hosts. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- “[Where to Specify Data Connector Options](#)” in [SAS Cloud Analytic Services: User’s Guide](#)
- “[DB2 Data Connector](#)” in [SAS Cloud Analytic Services: User’s Guide](#)

---

# LIBNAME Statement for the DB2 Engine under UNIX and PC Hosts

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to DB2 under UNIX and PC Hosts supports. For general information about this feature, see [LIBNAME Statement for Relational Databases](#) on page 127.

Here is the LIBNAME statement syntax for accessing DB2 under UNIX and PC Hosts.

**LIBNAME** *libref db2 <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

*db2*

specifies the SAS/ACCESS engine name for the DB2 under UNIX and PC Hosts interface.

*connection-options*

provides connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to DB2 several ways. Specify only one of these methods for each connection because they are mutually exclusive.

- USER=, PASSWORD=, DATASRC=
- COMPLETE=
- NOPROMPT=

- PROMPT=
- REQUIRED=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**USER=<'>DB2-user-name<'>**

lets you connect to a DB2 database with a user ID that is different from the default ID. USER= is optional. If you specify USER=, you must also specify PASSWORD=. If USER= is omitted, your default user ID for your operating environment is used.

Alias: UID=

**PASSWORD=<'>DB2-password<'>**

specifies the DB2 password that is associated with your DB2 user ID. PASSWORD= is optional. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. If you specify USER=, you must specify PASSWORD=.

Alias: PWD=

**DATASRC=<'>data-source-name<'>**

specifies the DB2 database to which you want to connect. DATASRC= is optional. If you omit it, you connect by using a default environment variable.

Alias: DATABASE=, DB=, DSN=

**COMPLETE=<'>CLI-connection-string<'>**

specifies connection information for your database for PCs only. Separate multiple options with a semicolon. When a successful connection is made, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, you are prompted with a dialog box that displays the values from the COMPLETE= connection string. You can edit any field before you connect to the database. See your DB2 documentation for more details.

This option is not available on UNIX platforms.

**NOPROMPT=<'>CLI-connection-string<'>**

specifies connection information for your database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned (no dialog box is displayed in SAS windowing environment).

**PROMPT=<'> CLI-connection-string<'>**

specifies connection information for your database for PCs only. Separate multiple options with a semicolon. When a successful connection is made, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately attempt to connect to the DBMS. Instead, it displays a dialog box in SAS windowing environment that contains the values that you entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the database.

Restriction: This option is not available on UNIX platforms.

**REQUIRED=<'>CLI-connection-string<'>**

specifies connection information for your database for PCs only. Separate the multiple options with semicolons. When a successful connection is made, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, a dialog box prompts you for the connection options. REQUIRED= lets you modify only required fields in the dialog box.

This option is not available on UNIX platforms.

***LIBNAME-options***

specifies how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to DB2 under UNIX and PC Hosts, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 18.1** SAS/ACCESS LIBNAME Options for DB2 under UNIX and PC Hosts

| Option               | Default Value                                                                       | Valid in CONNECT |
|----------------------|-------------------------------------------------------------------------------------|------------------|
| ACCESS=              | none                                                                                |                  |
| AUTHDOMAIN=          | none                                                                                |                  |
| AUTOCOMMIT=          | NO                                                                                  | •                |
| BL_RECOVERABLE=      | NO                                                                                  |                  |
| CONNECTION=          | SHAREDREAD                                                                          | •                |
| CONNECTION_GROUP=    | none                                                                                | •                |
| CURSOR_TYPE=         | none                                                                                | •                |
| DBCLIENT_MAX_BYTES=  | 1                                                                                   | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows), 10,000 (when bulk loading rows) |                  |
| DBCONINIT=           | none                                                                                | •                |
| DBCONTERM=           | none                                                                                | •                |
| DBCREATE_TABLE_OPTS= | none                                                                                |                  |
| DBGEN_NAME=          | DBMS                                                                                | •                |

| Option                     | Default Value                                                  | Valid in CONNECT |
|----------------------------|----------------------------------------------------------------|------------------|
| DBINDEX=                   | NO                                                             |                  |
| DBLIBINIT=                 | none                                                           |                  |
| DBLIBTERM=                 | none                                                           |                  |
| DBMAX_TEXT=                | 1024                                                           | •                |
| DBMSTEMP=                  | NO                                                             |                  |
| DBNULLKEYS=                | YES                                                            |                  |
| DBNULLWHERE=               | YES                                                            |                  |
| DBPROMPT=                  | NO                                                             | •                |
| DBSERVER_MAX_BYTES=        | none                                                           | •                |
| DBSLICEPARM=               | NONE                                                           |                  |
| DEFER=                     | NO                                                             | •                |
| DIRECT_EXE=                | none                                                           |                  |
| DIRECT_SQL=                | YES                                                            |                  |
| FETCH-IDENTITY=            | NO                                                             |                  |
| IGNORE_READ_ONLY_COLUMNS = | NO                                                             |                  |
| IN=                        | none                                                           |                  |
| INSERTBUFF=                | automatically calculated based on row length                   |                  |
| POST_STMT_OPTS=            | none                                                           |                  |
| MULTI_DATASRC_OPT=         | NONE                                                           |                  |
| PRESERVE_COL_NAMES=        | YES (see “Naming Conventions for DB2 under UNIX and PC Hosts”) |                  |
| PRESERVE_COMMENTS=         | NO                                                             | •                |

| Option                  | Default Value                                                   | Valid in CONNECT |
|-------------------------|-----------------------------------------------------------------|------------------|
| PRESERVE_TAB_NAMES=     | YES (see “Naming Conventions for DB2 under UNIX and PC Hosts”)  |                  |
| PRESERVE_USER=          | NO                                                              | •                |
| PROGRAM_NAME=           | none                                                            | •                |
| QUERY_TIMEOUT=          | 0                                                               | •                |
| READBUFF=               | automatically calculated based on row length                    | •                |
| READ_ISOLATION_LEVEL=   | CS                                                              |                  |
| READ_LOCK_TYPE=         | ROW                                                             | •                |
| REREAD_EXPOSURE=        | NO                                                              | •                |
| SCHEMA=                 | your user ID                                                    |                  |
| SPOOL=                  | YES                                                             |                  |
| SQL_FUNCTIONS=          | none                                                            |                  |
| SQL_FUNCTIONS_COPY=     | none                                                            |                  |
| SQLGENERATION=          | none                                                            |                  |
| STRINGDATES=            | NO                                                              | •                |
| UPDATE_ISOLATION_LEVEL= | CS (see “Locking in the DB2 under UNIX and PC Hosts Interface”) |                  |
| UPDATE_LOCK_TYPE=       | ROW                                                             | •                |
| UTILCONN_TRANSIENT=     | YES                                                             |                  |
| WARN_BIGINT=            | NO                                                              | •                |

---

## DB2 under UNIX and PC Hosts LIBNAME Statement Example

In this example, the libref MyDBLib uses the DB2 engine and the NOPROMPT= option to connect to a DB2 database. PROC PRINT is used to display the contents of the DB2 table Customers.

```
libname mydblib db2
    noprompt="dsn=userdsn;uid=myusr1;pwd=mypwd1";

proc print data=mydblib.customers;
    where state='CA';
run;
```

---

## Data Set Options for DB2 under UNIX and PC Hosts

All SAS/ACCESS data set options in this table are supported for DB2 under UNIX and PC Hosts. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 18.2** SAS/ACCESS Data Set Options for DB2 under UNIX and PC Hosts

| Option                 | Default Value                                                           |
|------------------------|-------------------------------------------------------------------------|
| BL_ALLOW_READ_ACCESS=  | NO                                                                      |
| BL_ALLOW_WRITE_ACCESS= | NO                                                                      |
| BL_CODEPAGE=           | the code page ID for the window                                         |
| BL_COPY_LOCATION=      | none                                                                    |
| BL_CPU_PARALLELISM=    | none                                                                    |
| BL_DATA_BUFFER_SIZE=   | none                                                                    |
| BL_DATAFILE=           | the temporary directory that is specified by the UTILLOC= system option |
| BL_DELETE_DATAFILE=    | YES                                                                     |
| BL_DISK_PARALLELISM=   | none                                                                    |

| <b>Option</b>        | <b>Default Value</b>                                                    |
|----------------------|-------------------------------------------------------------------------|
| BL_EXCEPTION=        | none                                                                    |
| BL_INDEXING_MODE=    | AUTOSELECT                                                              |
| BL_LOAD_REPLACE=     | NO                                                                      |
| BL_LOG=              | the temporary directory that is specified by the UTILLOC= system option |
| BL_METHOD=           | none                                                                    |
| BL_OPTIONS=          | none                                                                    |
| BL_PORT_MAX=         | none                                                                    |
| BL_PORT_MIN=         | none                                                                    |
| BL_RECOVERABLE=      | NO                                                                      |
| BL_REMOTE_FILE=      | none                                                                    |
| BL_SERVER_DATAFILE=  | the temporary directory that is specified by the UTILLOC= system option |
| BL_WARNING_COUNT=    | 2147483646                                                              |
| BULKLOAD=            | NO                                                                      |
| CHAR_AS_BINARY=      | NO                                                                      |
| CURSORTYPE=          | LIBNAME option value                                                    |
| DBCOMMIT=            | LIBNAME option value                                                    |
| DBCONDITION=         | none                                                                    |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                                                    |
| DBFORCE=             | NO                                                                      |
| DBGEN_NAME=          | DBMS                                                                    |
| DBINDEX=             | LIBNAME option value                                                    |
| DBKEY=               | none                                                                    |
| DBLABEL=             | NO                                                                      |

| Option                    | Default Value                                                      |
|---------------------------|--------------------------------------------------------------------|
| DBLARGETABLE=             | none                                                               |
| DBMAX_TEXT=               | 1024                                                               |
| DBNULL=_ALL_=YES          |                                                                    |
| DBNULLKEYS=               | LIBNAME option value                                               |
| DBNULLWHERE=              | LIBNAME option value                                               |
| DBPROMPT=                 | LIBNAME option value                                               |
| DBSASLABEL=               | COMPAT                                                             |
| DBSASTYPE=                | see “ <a href="#">Data Types for DB2 under UNIX and PC Hosts</a> ” |
| DBSLICE=                  | none                                                               |
| DBSLICEPARM=              | NONE                                                               |
| DBTYPE=                   | see “ <a href="#">Data Types for DB2 under UNIX and PC Hosts</a> ” |
| ERRLIMIT=                 | 1                                                                  |
| FETCH_IDENTITY=           | NO                                                                 |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                                 |
| IN=                       | LIBNAME option value                                               |
| INSERTBUFF=               | LIBNAME option value                                               |
| NULLCHAR=                 | SAS                                                                |
| NULLCHARVAL=              | a blank character                                                  |
| POST_STMT_OPTS=           | none                                                               |
| POST_TABLE_OPTS=          | none                                                               |
| PRE_STMT_OPTS=            | none                                                               |
| PRE_TABLE_OPTS=           | none                                                               |
| PRESERVE_COL_NAMES=       | LIBNAME option value                                               |

| Option                  | Default Value        |
|-------------------------|----------------------|
| QUERY_TIMEOUT=          | LIBNAME option value |
| READ_ISOLATION_LEVEL=   | LIBNAME option value |
| READ_LOCK_TYPE=         | LIBNAME option value |
| READBUFF=               | LIBNAME option value |
| SASDATEFMT=             | none                 |
| SCHEMA=                 | LIBNAME option value |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |
| UPDATE_LOCK_TYPE=       | LIBNAME option value |

---

## SQL Pass-Through Facility Specifics for DB2 under UNIX and PC Hosts

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the DB2 under UNIX and PC Hosts interface.

- The *dbms-name* is **DB2**.
- The CONNECT statement is required.
- You can connect to only one DB2 database at a time. However, you can use multiple CONNECT statements to connect to multiple DB2 databases by using the *alias* argument to distinguish your connections.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).
- These LIBNAME options are available with the CONNECT statement.
  - [AUTOCOMMIT=](#)
  - [CURSOR\\_TYPE=](#)
  - [QUERY\\_TIMEOUT=](#)

- [READBUFF=](#)
- [READ\\_ISOLATION\\_LEVEL=](#)

## Examples

This example connects to the SAMPLE database and sends it two EXECUTE statements to process.

```
proc sql;
  connect to db2 (database=sample);
  execute (create view
    sasdemo.whotookorders as
    select ordernum, takenby,
      firstname, lastname, phone
    from sasdemo.orders,
      sasdemo.employees
    where sasdemo.orders.takenby=
      sasdemo.employees.empid)
  by db2;
  execute (grant select on
    sasdemo.whotookorders to myusr1)
  by db2;
  disconnect from db2;
quit;
```

This example connects to the SAMPLE database by using an alias (DB1) and performs a query, shown in italic type, on the SASDEMO.CUSTOMERS table.

```
proc sql;
  connect to db2 as db1 (database=sample);
  select *
  from connection to db1
  (select * from sasdemo.customers
   where customer like '1%');
  disconnect from db1;
quit;
```

## Special Catalog Queries

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts supports the following special queries. You can use the queries to call the ODBC-style catalog function application programming interfaces (APIs). Here is the general format of these queries:

**DB2::SQLAPI** “*parameter 1*”,“*parameter n*”

**DB2::**  
is required to distinguish special queries from regular queries.

**SQLAPI**  
is the specific API that is being called. Neither DB2:: nor SQLAPI are case sensitive.

*"parameter n"*

is a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, this call to SQLTables usually matches table names such as mytest and my\_test:

```
select * from connection to db2 (DB2::SQLTables "test","","my_test");
```

Use the escape character to search only for the my\_test table:

```
select * from connection to db2 (DB2::SQLTables "test","","my\_test");
```

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts supports these special queries:

**DB2::SQLDataSources**

returns a list of database aliases that have been cataloged on the DB2 client.

**DB2::SQLDBMSInfo**

returns information about the DBMS server and version. It returns one row with two columns that describe the DBMS name (such as DB2/NT) and version (such as 8.2).

# Autopartitioning Scheme for DB2 under UNIX and PC Hosts

## Overview

Autopartitioning for SAS/ACCESS Interface to DB2 for UNIX and PC Hosts is a modulo (MOD) function method. For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

## Autopartitioning Restrictions

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- INTEGER and SMALLINT columns are given preference.
- You can use other DB2 numeric columns for partitioning as long as the precision minus the scale of the column is between 0 and 10—namely,  $0 < (precision - scale) < 10$ .

---

## Nullable Columns

If you select a nullable column for autopartitioning, the `OR<column-name> IS NULL` SQL statement is appended at the end of the SQL code that is generated for the threaded Reads. This ensures that any possible NULL values are returned in the result set.

---

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in a SAS WHERE clause. For example, this DATA step cannot use a threaded Read to retrieve data because all numeric columns in the table (see the table definition in “[Using DBSLICE=](#)”) are in the WHERE clause:

```
data work.locemp;
  set trlib.MYEMPS;
  where EMPNUM<=30 and ISTENURE=0 and
        SALARY<=35000 and NUMCLASS>2;
run;
```

---

## Using DBSLICEPARM=

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts defaults to three threads when you use autopartitioning. However, do not specify a maximum number of threads for the threaded Read in the `DBSLICEPARM= LIBNAME` option.

---

## Using DBSLICE=

You might achieve the best possible performance when using threaded Reads by specifying the `DBSLICE= data set option` for DB2 in your SAS operation. This is especially true if your DB2 data is evenly distributed across multiple partitions in a DB2 Enterprise Extended Edition (EEE) database system. When you create a DB2 table under the DB2 EEE model, you can specify the partitioning key that you want to use by appending the clause `PARTITIONING KEY(column-name)` to your CREATE TABLE statement. Here is how you can accomplish this by using the LIBNAME option, `DBCREATE_TABLE_OPTS=`, within the SAS environment.

```
/*points to a triple node server*/
libname trlib2 db2 user=db2user pw="db2pwd" datasrc=sample3c
DBCREATE_TABLE_OPTS='PARTITIONING KEY(EMPNUM);

proc datasets library=trlib;
  delete MYEMPS1;run;
```

```

data trlib.myemps(drop=morf whatstate
      DBTYPE=(HIREDATE="date" SALARY="numeric(8,2)"
      NUMCLASS="smallint" GENDER="char(1)" ISTENURE="numeric(1)"
      STATE="char(2)"
      EMPNUM="int NOT NULL Primary Key"));
format HIREDATE mmddyy10.;
do EMPNUM=1 to 100;
      morf=mod(EMPNUM,2)+1;
      if(morf eq 1) then
            GENDER='F';
      else
            GENDER='M';
      SALARY=(ranuni(0)*5000);
      HIREDATE=int(ranuni(13131)*3650);
      whatstate=int(EMPNUM/5);
      if(whatstate eq 1) then
            STATE='FL';
      if(whatstate eq 2) then
            STATE='GA';
      if(whatstate eq 3) then
            STATE='SC';
      if(whatstate eq 4) then
            STATE='VA';
      else
            state='NC';
      ISTENURE=mod(EMPNUM,2);
      NUMCLASS=int(EMPNUM/5)+2;
      output;
end;
run;

```

After the table MYEMPS is created on this three-node database, a third of the rows reside on each of the three nodes.

Optimization of the threaded Read against this partitioned table depends on the location of the DB2 partitions. If the DB2 partitions are on the same machine, you can use **DBSLICE=** with the DB2 NODENUMBER function in the WHERE clause:

```

proc print data=trlib2.MYEMPS(DBSLICE= ("NODENUMBER(EMPNO)=0"
      "NODENUMBER(EMPNO)=1" "NODENUMBER(EMPNO)=2") );
run;

```

If the DB2 partitions reside on different physical machines, you can usually obtain the best results by using the DBSLICE= option with the SERVER= syntax in addition to the DB2 NODENUMBER function in the WHERE clause.

In the next example, DBSLICE= contains specific partitioning information for DB2. Also, Sample3a, Sample3b, and Sample3c are DB2 database aliases that point to individual DB2 EEE database nodes that exist on separate physical machines. For more information about the configuration of these nodes, see ["Configuring DB2 EEE Nodes on Physically Partitioned Databases" on page 764](#).

```

proc print data=trlib2.MYEMPS(DBSLICE=(sample3a="NODENUMBER(EMPNO)=0"
      sample3b="NODENUMBER(EMPNO)=1" sample3c="NODENUMBER(EMPNO)=2") );
run;

```

NODENUMBER is not required to use threaded Reads for SAS/ACCESS Interface to DB2 under UNIX and PC Hosts. The methods and examples described in [DBSLICE= on page 531](#) work well in cases where the table that you want to read is not stored in multiple partitions to DB2. These methods also give you full control

over which column is used to execute the threaded Read. For example, if the STATE column in your employee table contains only a few distinct values, you can modify your DBSLICE= clause accordingly:

```
data work.locemp;
  set trllib2.MYEMPS (DBSLICE= ("STATE='GA'"
                                "STATE='SC'" "STATE='VA'" "STATE='NC'"));
  where EMPNUM<=30 and ISTEMURE=0 and SALARY<=35000 and NUMCLASS>2;
run;
```

## Configuring DB2 EEE Nodes on Physically Partitioned Databases

Assuming that the database SAMPLE is partitioned across three different machines, you can create a database alias for it at each node from the DB2 Command Line Processor by issuing these commands:

```
catalog tcpip node node1 remote <hostname> server 50000
catalog tcpip node node2 remote <hostname> server 50000
catalog tcpip node node3 remote <hostname> server 50000
catalog database sample as samplea at node node1
catalog database sample as sampleb at node node2
catalog database sample as samplec at node node3
```

This enables SAS/ACCESS Interface to DB2 to access the data for the SAMPLE table directly from each node. For more information about configuring DB2 EEE to use multiple physical partitions, see the *DB2 Administrator's Guide*.

## Temporary Table Support for DB2 under UNIX and PC Hosts

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

---

# Calling Stored Procedures in DB2 under UNIX and PC Hosts

---

## Overview

A stored procedure is one or more SQL statements or supported third-generation languages (3GLs, such as C) statements that are compiled into a single procedure that exists in DB2. Stored procedures might contain static (hardcoded) SQL statements. Static SQL is optimized better for some DBMS operations. In a carefully managed DBMS environment, programmers and database administrators can know the exact SQL to execute.

SAS usually generates SQL dynamically. However, database administrators can encode static SQL in a stored procedure and therefore restrict SAS users to a tightly controlled interface. When you use a stored procedure call, you must specify a schema.

SAS/ACCESS support for stored procedure includes passing input parameters, retrieving output parameters into SAS macro variables, and retrieving the result set into a SAS table. Although DB2 stored procedures can return multiple result sets, SAS/ACCESS Interface to DB2 under UNIX and PC Hosts can retrieve only a single result set.

You can call stored procedures only from PROC SQL.

---

## Examples

---

### Example 1: Specify a Basic Call

Use CALL statement syntax to call a stored procedure.

```
call "schema".stored_proc
```

The simplest way to call a stored procedure is to use the EXECUTE statement in PROC SQL. In this example, you execute STORED\_PROC by using a CALL statement. SAS does not capture the result set.

```
proc sql;
  connect to db2 (db=sample uid= pwd=);
  execute (call "schema".stored_proc);
  quit;
```

---

## Example 2: Specify One Input Parameter That Returns a Result Set

You can also return a result set. In this example, MYPROC3 is executed using a CALL statement and returns a result set.

```
proc sql;
  connect to db2 (db=sample uid= pwd=);
  %let p1 = 2000;
  select * from connection to db2 (call MYSCHHEMA.MYPROC3 (:p1));
```

---

## Example 3: Specify Three Output Parameters

The CALL statement syntax supports passing of parameters. You can specify such input parameters as numeric constants, character constants, or a null value. You can also pass input parameters by using SAS macro variable references. To capture the value of an output parameter, a SAS macro variable reference is required. This example uses a constant (1), an input/output parameter (:INOUT), and an output parameters (:OUT). Not only is the result set returned to the SAS results table, the SAS macro variables INOUT and OUT capture the parameter outputs.

```
proc sql;
  connect to db2 (db=sample uid= pwd=);
  %let INOUT=2;
  create table sasresults as select * from connection to db2
    (call "schema".stored_proc (1,:INOUT,:OUT));
  quit;
```

---

## Example 4: Specify Three Different Parameters

The CALL statement syntax supports passing of parameters. To capture the value of an output parameter, a SAS macro variable reference is required. This example uses three output parameters (:p1, :p2, :p3:) and displays the value of each.

```
proc sql;
  connect to db2 (db=sample uid= pwd=);
  execute (call MYSCHHEMA.MYPROC(:p1,:p2,:p3)) by db2;
  %put &p1 &p2 &p3 /* display values of output parameters */
```

---

## Example 5: Pass a NULL Parameter

In these calls, NULL is passed as the parameter to the DB2 stored procedure.

- null string literals in the call

```
call proc(' ');
call proc("")
```

- literal period or literal NULL in the call

```
call proc(.)
call proc(NULL)
```

- SAS macro variable set to NULL string

```
%let charparm=;
call proc(:charparm)
```

- SAS macro variable set to period (SAS numeric value is missing)

```
%let numparm=. ;
call proc(:numparm)
```

Only the literal period and the literal NULL work generically for both DB2 character parameters and DB2 numeric parameters. For example, a DB2 numeric parameter would reject "" and %let numparm=. ; would not pass a DB2 NULL for a DB2 character parameter. As a literal, a period passes NULL for both numeric and character parameters. However, it constitutes a NULL only for a DB2 numeric parameter when it is in a SAS macro variable.

You cannot pass NULL parameters by omitting the argument. For example, you cannot use this call to pass three NULL parameters.

```
call proc(,,)
```

You could use this call instead.

```
call proc(NULL,NULL,NULL)
```

---

## Example 6: Specify a Schema

Use standard CALL statement syntax to execute a stored procedure that exists in another schema, as shown in this example.

```
proc sql;
  connect to db2 (db=sample uid= pwd=);
  execute (call OTHERSCHEMA.MYPROC1(:p1));
quit;
```

If the schema is in mixed case or lowercase, enclose the schema name in double quotation marks.

```
proc sql;
  connect to db2 (db=sample uid= pwd=);
  execute (call "OTHERSCHEMA".MYPROC1(:p1));
quit;
```

---

## Example 7: Execute Remote Stored Procedures

If the stored procedure exists on a different DB2 instance, specify it with a valid three-part name.

```
select * from connection
proc sql;
connect to db2 (db=sample uid= pwd=);
to db2 (call MYSCHHEMA.MYPROC1.prod5(:p1));
quit;
```

---

## DBLOAD Procedure Specifics for DB2 under UNIX and PC Hosts

---

### Key Information

For general information about this feature, see [Appendix 3, “DBLOAD Procedure,” on page 1455](#).

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts supports all **DBLOAD** procedure statements in batch mode.

---

**Note:** SAS still supports this legacy procedure. However, to access your DBMS data more directly the best practice is to use the LIBNAME statement for DB2 under UNIX or PC Hosts or use the SQL pass-through facility. For more information, see [“LIBNAME Statement for the DB2 Engine under UNIX and PC Hosts” on page 751](#) and [“SQL Pass-Through Facility Specifics for DB2 under UNIX and PC Hosts” on page 759](#).

---

Here are the DBLOAD procedure specifics for the DB2 under UNIX and PC Hosts interface.

- DBMS= value is DB2.
- Here are the database description statements that PROC DBLOAD uses:

**IN= <'>database-name<'>;**

specifies the name of the database in which you want to store the new DB2 table. The IN= statement is required and must immediately follow the PROC DBLOAD statement. The *database-name* is limited to eight characters. DATABASE= is an alias for the IN= statement.

The database that you specify must already exist. If the database name contains the \_, \$, @, or # special character, you must enclose it in quotation

marks. DB2 recommends against using special characters in database names, however.

**USER= <'>user name<'>;**

lets you connect to a DB2 database with a user ID that is different from the default login ID.

USER= is optional in SAS/ACCESS Interface to DB2 under UNIX and PC Hosts. If you specify USER=, you must also specify PASSWORD=. If USER= is omitted, your default user ID is used.

**PASSWORD= <'>password<'>;**

specifies the password that is associated with your user ID.

PASSWORD= is optional in SAS/ACCESS Interface to DB2 under UNIX and PC Hosts because users have default user IDs. If you specify USER=, however, you must specify PASSWORD=.

If you do not wish to enter your DB2 password in uncoded text on this statement, see PROC PWENCODE in *Base SAS Procedures Guide* for a method to encode it.

■ Here is the TABLE= statement:

**TABLE= <'><schema-name.>table-name<'>;**

identifies the DB2 table or DB2 view that you want to use to create an access descriptor. The *table-name* is limited to 18 characters. If you use quotation marks, the name is case sensitive. The TABLE= statement is required.

The *schema-name* is a person's name or group ID that is associated with the DB2 table. The schema name is limited to eight characters.

■ Here is the NULLS statement.

**NULLS variable-identifier-1 =Y|N|D < . . . variable-identifier-n =Y|N|D >;**

lets you specify whether the DB2 columns that are associated with the listed SAS variables allow NULL values. By default, all columns accept NULL values.

The NULLS statement accepts any one of these values.

**Y**

specifies that the column accepts NULL values. This is the default.

**N**

specifies that the column does not accept NULL values.

**D**

specifies that the column is specified as NOT NULL WITH DEFAULT.

## Examples

This example creates a new DB2 table, SASDEMO.EXCHANGE, from the MYDBLIB.RATEOFEX data file. You must be granted the appropriate privileges in order to create new DB2 tables or views.

```
proc dbload dbms=db2 data=mydblib.rateofex;
  in='sample';
  user='myusr1';
  password=mypwd1;
```

```

table=sasdemo.exchange;
  rename fgnindol=fgnindollars
    4=dollarsinfgn;
  nulls updated=n fgnindollars=n
    dollarsinfgn=n country=n;
  load;
run;

```

This example sends only a DB2 SQL GRANT statement to the SAMPLE database and does not create a new table. Therefore, the TABLE= and LOAD statements are omitted.

```

proc dbload dbms=db2;
  in='sample';
  sql grant select on sasdemo.exchange
    to myusr1;
run;

```

## Passing SAS Functions to DB2 under UNIX and PC Hosts

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts passes the following SAS functions to DB2 for processing if the DBMS driver or client that you are using supports this function. Where the DB2 function name differs from the SAS function name, the DB2 name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                   |                      |
|-------------------|----------------------|
| ABS               | MIN                  |
| ARCOS (ACOS)      | MINUTE               |
| ARSIN (ASIN)      | MOD (see note)       |
| ATAN              | MONTH                |
| AVG               | QTR (QUARTER)        |
| BYTE (CHAR)       | REPEAT               |
| CEIL (CEILING)    | SECOND               |
| COS               | SIGN                 |
| COSH              | SIN                  |
| COT               | SINH                 |
| COUNT (COUNT_BIG) | SQRT                 |
| DAY (DAYOFMONTH)  | STRIP (RTRIM, LTRIM) |
| EXP               | SUBSTR (SUBSTRING)   |
| FLOOR             | SUM                  |
| HOUR              | TAN                  |
| INDEX (LOCATE)    | TANH                 |
| LENGTH            | TRANWRD (REPLACE)    |
| LOG               | TRIMN (RTRIM)        |
| LOG10             | UPCASE (UCASE)       |

|                 |                     |
|-----------------|---------------------|
| LOWCASE (LCASE) | WEEKDAY (DAYOFWEEK) |
| MAX             | YEAR                |

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to DB2. Due to incompatibility in date and time functions between DB2 and SAS, DB2 might not process them correctly. Check your results to determine whether these functions are working as expected.

|                    |                 |
|--------------------|-----------------|
| COMPRESS (REPLACE) | SOUNDEX         |
| DATE (CURDATE)     | TIME (CURTIME)  |
| DATEPART (DATE)    | TIMEPART        |
| DATETIME (NOW)     | TODAY (CURDATE) |

---

## Passing Joins to DB2 under UNIX and PC Hosts

For a multiple-libref join to pass to DB2, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- update isolation level (UPDATE\_ISOLATION\_LEVEL=, if specified)
- read\_isolation level (READ\_ISOLATION\_LEVEL=, if specified)
- qualifier (QUALIFIER=)
- data source (DATASRC=)
- prompt (PROMPT= must *not* be specified)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Sorting Data That Contains NULL Values

Some DBMSs, including DB2 under UNIX and PC Hosts, place NULL values at the end of a sorted list. SAS normally places NULL values at the beginning of a sorted list. In SAS, to use BY-group processing, SAS expects that values for a BY group are already sorted before it performs BY-group processing. If SAS determines that values are not sorted, then BY-group processing stops. Therefore, if NULL values are not at the beginning of a sorted list, SAS determines that the values are not sorted correctly.

If you use PROC SQL with an ORDER BY statement to sort your data, then the SQL code is generated so that NULL values are placed at the beginning of the sorted list. In this way, SAS can perform any BY-group processing without stopping.

For more information, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43 and “[Sorting DBMS Data](#)” on page 49.

---

## Bulk Loading for DB2 under UNIX and PC Hosts

---

### Overview

Bulk loading is the fastest way to insert large numbers of rows into a DB2 table. Using this facility instead of regular SQL insert statements, you can insert rows two to ten times more rapidly. DB2 bulk-load [examples](#) and tips for [maximizing performance](#) on page 775 are available. You must specify `BULKLOAD=YES` to use the bulk-load facility.

SAS/ACCESS Interface to DB2 under UNIX and PC Hosts offers LOAD, IMPORT, and CLI LOAD bulk-loading methods. The `BL_REMOTE_FILE=` and `BL_METHOD=` data set options determine which method to use.

For more information about the differences between IMPORT, LOAD, and CLI LOAD, see the *DB2 Data Movement Utilities Guide and Reference*.

---

### Using the LOAD Method

To use the LOAD method, you must have system administrator, database administrator, or load authority on the database and the INSERT privilege on the table to be loaded.

This method also requires that client and server machines can read and write files to a common location such as a mapped network drive or an NFS directory. To use this method, specify the `BL_REMOTE_FILE=` option.

**Note:** Because SAS/ACCESS Interface to DB2 uses the PC/IXF file format to transfer data to the DB2 LOAD utility, you cannot use this method to load data into partitioned databases.

---

## Data Set Options with Bulk Loading

Here are the bulk-load options available with the LOAD method. For details about these options, see [Data Set Options for Relational Databases on page 370](#).

- [BL\\_CODEPAGE=](#)
- [BL\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_LOG=](#): The log file contains a summary of load information and error descriptions. On most platforms, the default file name is `BL_<table>_<unique-ID>.log`, where *table* specifies the table name and *unique-ID* specifies a number used to prevent collisions in the event of two or more simultaneous instances of bulk loading of a particular table. The SAS/ACCESS engine generates this number.
- [BL\\_OPTIONS=](#)
- [BL\\_REMOTE\\_FILE=](#)
- [BL\\_SERVER\\_DATAFILE=](#)
- [BL\\_WARNING\\_COUNT=](#)

---

## Using the IMPORT Method

The IMPORT method does not offer the same level of performance as the LOAD method. However, it is available to all users with INSERT privileges for the tables to be loaded. The IMPORT method does not require that the server and client have a common location in order to access the data file. If you do not specify `BL_REMOTE_FILE=`, the IMPORT method is automatically used.

Here are the bulk-load options available with the IMPORT method. For detailed information about these options, see [“Overview” on page 370](#).

- [BL\\_CODEPAGE=](#)
- [BL\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_LOG=](#)
- [BL\\_OPTIONS=](#)

---

## Using the CLI LOAD Method

The CLI LOAD method is an interface to the standard DB2 LOAD utility, which gives the added performance of using LOAD but without specifying additional options for bulk loading. This method requires the same privileges as the LOAD method and is available only in DB2 Version 7 FixPak 4 and later clients and servers. If your client and server can support the CLI LOAD method, you can generally see the best performance by using it. The CLI LOAD method can also be used to load data into a partitioned DB2 database for client and database nodes that are DB2 Version 8.1 or later. To use this method, specify BL\_METHOD=CLLOAD as a data set option. Here are the bulk-load options that are available with the CLI LOAD method:

- [BL\\_ALLOW\\_READ\\_ACCESS=](#)
- [BL\\_ALLOW\\_WRITE\\_ACCESS=](#)
- [BL\\_COPY\\_LOCATION=](#)
- [BL\\_CPU\\_PARALLELISM=](#)
- [BL\\_DATA\\_BUFFER\\_SIZE=](#)
- [BL\\_DISK\\_PARALLELISM=](#)
- [BL\\_EXCEPTION](#)
- [BL\\_INDEXING\\_MODE=](#)
- [BL\\_LOAD\\_REPLACE=](#)
- [BL\\_LOG=](#)
- [BL\\_METHOD=](#)
- [BL\\_OPTIONS=](#)
- [BL\\_RECOVERABLE=](#)
- [BL\\_REMOTE\\_FILE=](#)

---

## Capturing Bulk-Load Statistics into Macro Variables

These bulk-load macro variables capture how many rows are loaded, skipped, rejected, committed, and deleted before writing this information to the SAS log.

- [SYSBL\\_ROWSCOMMITTED](#)
- [SYSBL\\_ROWSDELETED](#)
- [SYSBL\\_ROWSLOADED](#)
- [SYSBL\\_ROWSREJECTED](#)
- [SYSBL\\_ROWSSKIPPED](#)

---

## Maximizing Load Performance for DB2 under UNIX and PC Hosts

These tips can help you optimize LOAD performance when you are using the DB2 bulk-load facility.

- Specifying `BL_REMOTE_FILE=` causes the loader to use the DB2 LOAD utility. This is much faster than the IMPORT utility, but it requires database administrator authority.
- Performance might suffer if the value for `DBCOMMIT=` is too low. Increase the default (10000 when `BULKLOAD=YES`) for improved performance.
- Increasing the DB2 tuning parameters, such as Utility Heap and input-output characteristics, improves performance. These parameters are controlled by your database or server administrator.
- When using the IMPORT utility, specify `BL_OPTIONS="COMPOUND=x"`—where x is a number between 1 and 7 on Windows, and between 1 and 100 on UNIX. This causes the IMPORT utility to insert multiple rows for each execute instead of one row per execute.
- When using the LOAD utility on a multi-processor or multi-node DB2 server, specify `BL_OPTIONS="ANYORDER"` to improve performance. This might cause DB2 log entries to be out of order because it lets DB2 insert rows in a different order from how they appear in the loader data file.

---

## Examples

This example shows how to use a SAS data set, `SASFDT.FLT98`, to create and load a large DB2 table, `FLIGHTS98`. Because the code specifies `BULKLOAD=YES` and `BL_REMOTE_FILE=` is omitted, this load uses the DB2 IMPORT command.

```
libname sasflt 'SAS-library';
libname db2_air db2 user=myuser using=mypwd
      database='db2_flt' schema=statsdiv;

proc sql;
  create table db2_air.flights98
    (bulkload=YES bl_options='compound=7 norowwarnings')
    as select * from sasflt.flt98;
quit;
```

The `BL_OPTIONS=` option passes DB2 file type modifiers to DB2. The `norowwarnings` modifier indicates that all row warnings about rejected rows are to be suppressed.

This example shows how to append the SAS data set, `SASFDT.FLT98` to a pre-existing DB2 table, `ALLFLIGHTS`. Because the code specifies `BULKLOAD=YES` and `BL_REMOTE_FILE=`, this load uses the DB2 LOAD command.

```
proc append base=db2_air.allflights
```

```
(BULKLOAD=YES
  BL_REMOTE_FILE='/tmp/tmpflt'
  BL_LOG='/tmp/fltdata.log'
  BL_DATAFILE='/nfs/server/tmp/fltdata.ixf'
  BL_SERVER_DATAFILE='/tmp/fltdata.ixf')
data=sasflt.flt98;
run;
```

Here, BL\_REMOTE\_FILE= and BL\_SERVER\_DATAFILE= are paths relative to the server. BL\_LOG= and BL\_DATAFILE= are paths relative to the client.

This example shows how to use the SAS data set SASFLT.ALLFLIGHTS to create and load a large DB2 table, ALLFLIGHTS. Because the code specifies BULKLOAD=YES and BL\_METHOD=CLILOAD, this operation uses the DB2 CLI LOAD interface to the LOAD command.

```
data db2_air.allflights(BULKLOAD=YES BL_METHOD=CLILOAD);
set sasflt.allflights;
run;
```

## Locking in the DB2 under UNIX and PC Hosts Interface

These LIBNAME and data set options let you control how the DB2 under UNIX and PC Hosts interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134. For additional information, see your DB2 documentation.

**READ\_LOCK\_TYPE=** ROW | TABLE

**UPDATE\_LOCK\_TYPE=** ROW | TABLE

**READ\_ISOLATION\_LEVEL=** RR | RS | CS | UR

The DB2 database manager supports the RR, RS, CS, and UR isolation levels that are defined in the following table. Regardless of the isolation level, the database manager places exclusive locks on every row that is inserted, updated, or deleted. All isolation levels therefore ensure that only this application process can change any given row during a unit of work: No other application process can change any rows until the unit of work is complete.

**Table 18.3** Isolation Levels for DB2 under UNIX and PC Hosts

| Isolation Level      | Definition                                                       |
|----------------------|------------------------------------------------------------------|
| RR (Repeatable Read) | no dirty Reads, no nonrepeatable Reads, no phantom Reads         |
| RS (Read Stability)  | no dirty Reads, no nonrepeatable Reads; does allow phantom Reads |

| Isolation Level       | Definition                                                       |
|-----------------------|------------------------------------------------------------------|
| CS (Cursor Stability) | no dirty Reads; does allow nonrepeatable Reads and phantom Reads |
| UR (Uncommitted Read) | allows dirty Reads, nonrepeatable Reads, and phantom Reads       |

Here is how the terms in the table are defined.

#### *Dirty reads*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it can see changes that those concurrent transactions made even before they commit them.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

#### *Nonrepeatable reads*

If a transaction exhibits this phenomenon, then the system read a row once and was unable to read the row again later in the same transaction. This might occur if a concurrent transaction changed or even deleted the row. Therefore, the read is not (necessarily) repeatable.

For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

#### *Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist (a "phantom").

`UPDATE_ISOLATION_LEVEL= CS | RS | RR`

The DB2 database manager supports the CS, RS, and RR isolation levels defined in the preceding table. Uncommitted reads are not allowed with this option.

---

# Naming Conventions for DB2 under UNIX and PC Hosts

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) DB2 is not case sensitive, and all names default to uppercase.

DB2 objects include tables, views, columns, and indexes. They follow these naming conventions.

- A name can begin with a letter or one of these symbols: dollar sign (\$), number or pound sign (#), or at symbol (@).
- A table name or a column name must be from 1 to 128 characters long.
- A name can contain the letters A to Z, any valid letter with a diacritic, numbers from 0 to 9, underscore (\_), dollar sign (\$), number or pound sign (#), or at symbol (@).
- Names are not case sensitive. For example, the table names CUSTOMER and Customer are the same, but object names are converted to uppercase when they are entered. If a name is enclosed in quotation marks, the name is case sensitive.
- A name cannot be a DB2 reserved word or an SQL-reserved word, such as WHERE or VIEW.
- A name cannot be the same as another DB2 object that has the same type.

Schema and database names have similar conventions, except that they are each limited to 30 and 8 characters respectively. For more information, see your DB2 SQL reference documentation.

---

## Data Types for DB2 under UNIX and PC Hosts

---

### Overview

Every column in a table has a name and a data type. The data type tells DB2 how much physical storage to set aside for the column and the form in which the data is stored. DB2 uses IBM SQL data types. This section includes information about DB2 data types, null and default values, and data conversions.

For more information about DB2 data types and to determine which data types are available for your version of DB2, see your DB2 SQL reference documentation.

---

# Supported Data Types for DB2 under UNIX and PC Hosts

Here are the data types that are supported for DB2 under UNIX and PC Hosts:

- Character data:

|                               |                        |
|-------------------------------|------------------------|
| CHAR( <i>n</i> )              | LONG VARGRAPHIC        |
| CLOB (character large object) | VARCHAR( <i>n</i> )    |
| GRAPHIC( <i>n</i> )           | VARGRAPHIC( <i>n</i> ) |
| LONG VARCHAR                  |                        |

- Numeric data:

|                                   |          |
|-----------------------------------|----------|
| BIGINT                            | INTEGER  |
| DECIMAL   DEC   NUMERIC   NUM     | SMALLINT |
| FLOAT   DOUBLE   DOUBLE PRECISION |          |

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

---

- Date and time data:

|      |           |
|------|-----------|
| DATE | TIMESTAMP |
| TIME |           |

---

**Note:** SQL date and time data types are collectively called datetime values. Be aware that columns of these data types can contain data values that are out of range for SAS.

---

- Binary data: BLOB (binary large object)

For more information about DB2 data types, see your DB2 documentation.

---

## Handling Increased Precision with TIMESTAMP Values

With the upgrade from DB2 9.5 to DB2 9.7, support was added for greater precision with TIMESTAMP values, increasing from 6 decimal values for a second up to 12 decimal values for a second. This value can be larger than the current SAS limit on

the precision of numeric values, which is how datetime values are stored. This increase in precision for DB2, combined with the current SAS limit on precision, can result in a SAS formatted value that incorrectly represents the last few decimal digits of a SAS datetime value.

To avoid this issue, you can use the `SAS_DB2_TS_REDUCE_SCALE` environment variable. When you specify `SAS_DB2_TS_REDUCE_SCALE=YES`, SAS/ACCESS Interface to DB2 defaults to a format length of 26 and a six decimal values for a second.

For more information, see “[Choosing Your Degree of Numeric Precision](#)” on page 8.

## DB2 Null and Default Values

DB2 has a special value called NULL. A DB2 NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a DB2 NULL value, it interprets it as a SAS missing value.

You can specify a column in a DB2 table so that it requires data. To do this in SQL, you specify a column as NOT NULL. NOT NULL tells SQL to allow a row to be added to a table only if there is a value for the field. For example, NOT NULL assigned to the field CUSTOMER in the table `SASDEMO.CUSTOMER` does not allow a row to be added unless there is a value for CUSTOMER. When creating a DB2 table with SAS/ACCESS, you can use the `DBNULL=` data set option to indicate whether NULL is a valid value for specified columns.

DB2 columns can also be specified as NOT NULL WITH DEFAULT. For more information about using the NOT NULL WITH DEFAULT value, see your DB2 SQL reference documentation.

Once you know whether a DB2 column enables NULLs or the host system provides a default value for a column that is defined as NOT NULL WITH DEFAULT, you can write selection criteria and enter values to update a table. Unless a column is specified as NOT NULL or NOT NULL WITH DEFAULT, it allows NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how the DBMS handles SAS missing character values, use the `NULCHAR=` and `NULCHARVAL=` data set options.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to DB2 assigns to SAS variables when using the `LIBNAME` statement to read from a DB2 table. These default formats are based on DB2 column attributes.

**Table 18.4 LIBNAME Statement: Default SAS Formats for DB2 Data Types**

| DB2 Data Type | SAS Data Type | Default SAS Format |
|---------------|---------------|--------------------|
| BLOB          | character     | \$HEXw.            |

| <b>DB2 Data Type</b>                | <b>SAS Data Type</b> | <b>Default SAS Format</b> |
|-------------------------------------|----------------------|---------------------------|
| CLOB                                | character            | \$w.                      |
| CHAR( <i>n</i> ) <sup>1</sup>       | character            | \$w.                      |
| VARCHAR( <i>n</i> ) <sup>1</sup>    |                      |                           |
| LONG VARCHAR                        |                      |                           |
| GRAPHIC( <i>n</i> ) <sup>1</sup>    | character            | \$w.                      |
| VARGRAPHIC( <i>n</i> ) <sup>1</sup> |                      |                           |
| LONG VARGRAPHIC                     |                      |                           |
| INTEGER                             | numeric              | 11.                       |
| SMALLINT                            | numeric              | 6.                        |
| BIGINT                              | numeric              | 20.                       |
| DECIMAL                             | numeric              | w.d                       |
| NUMERIC                             | numeric              | w.d                       |
| FLOAT                               | numeric              | none                      |
| DOUBLE                              | numeric              | none                      |
| TIME                                | numeric              | TIME8.                    |
| DATE                                | numeric              | DATE9.                    |
| TIMESTAMP                           | numeric              | DATETIMEw.d               |

<sup>1</sup> *n* in DB2 character and graphic data types is equivalent to *w* in SAS formats.

This table shows the default DB2 data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 18.5 LIBNAME Statement: Default DB2 Data Types for SAS Variable Formats**

| <b>SAS Variable Format</b> | <b>DB2 Data Type</b>                                                                                                 |
|----------------------------|----------------------------------------------------------------------------------------------------------------------|
| w.d                        | DECIMAL ( <i>m,n</i> ) <sup>2</sup>                                                                                  |
| other numerics             | DOUBLE                                                                                                               |
| \$w.                       | VARCHAR( <i>n</i> ) <sup>1</sup><br>( <i>n</i> <=4000)<br>LONG VARCHAR( <i>n</i> ) <sup>1</sup><br>( <i>n</i> >4000) |

| SAS Variable Format | DB2 Data Type |
|---------------------|---------------|
| datetime formats    | TIMESTAMP     |
| date formats        | DATE          |
| time formats        | TIME          |

- 1 *n* in DB2 data types is equivalent to *w* in SAS formats.  
 2 *m* and *n* in DB2 numeric data types are equivalent to *w* and *d* in SAS formats.

## DBLOAD Procedure Data Conversions

This table shows the default DB2 data types that SAS/ACCESS assigns to SAS variable formats when you use the [DBLOAD procedure](#).

*Table 18.6 PROC DBLOAD: Default DB2 Data Types for SAS Variable Formats*

| SAS Variable Format             | DB2 Data Type                      |
|---------------------------------|------------------------------------|
| \$ <i>w</i> .                   | CHAR( <i>n</i> ) <sup>1</sup>      |
| <i>w</i> .                      | DECIMAL( <i>p</i> ) <sup>2</sup>   |
| <i>w.d</i>                      | DECIMAL( <i>p,s</i> ) <sup>2</sup> |
| IB <i>w.d</i> , PIB <i>w.d</i>  | INTEGER                            |
| all other numerics <sup>3</sup> | DOUBLE                             |
| datetimew. <i>d</i>             | TIMESTAMP                          |
| date <i>w</i> .                 | DATE                               |
| time. <sup>4</sup>              | TIME                               |

- 1 *n* in DB2 character and graphic data types is equivalent to *w* in SAS formats.

- 2 *p* and *s* in DB2 numeric data types are equivalent to *w* and *d* in SAS formats.

- 3 Includes all SAS numeric formats, such as BINARY8 and E10.0.

- 4 Includes all SAS time formats, such as TOD*w,d* and HHMM*w,d*.

# Sample Programs for DB2 under UNIX and PC Hosts

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to DB2 under z/OS

---

|                                                                       |     |
|-----------------------------------------------------------------------|-----|
| <i>System Requirements for SAS/ACCESS Interface to DB2 under z/OS</i> | 787 |
| <i>Introduction to SAS/ACCESS Interface to DB2 under z/OS</i>         | 787 |
| <i>LIBNAME Statement for the DB2 Engine under z/OS</i>                | 788 |
| Overview                                                              | 788 |
| Arguments                                                             | 788 |
| DB2 under z/OS LIBNAME Statement Example                              | 791 |
| <i>Data Set Options for DB2 under z/OS</i>                            | 792 |
| <i>SQL Pass-Through Facility Specifics for DB2 under z/OS</i>         | 795 |
| Key Information                                                       | 795 |
| Examples                                                              | 795 |
| Special Catalog Queries                                               | 796 |
| <i>Autopartitioning Scheme for DB2 under z/OS</i>                     | 797 |
| Overview                                                              | 797 |
| Autopartitioning Restrictions                                         | 797 |
| Column Selection for MOD Partitioning                                 | 798 |
| How WHERE Clauses Restrict Autopartitioning                           | 798 |
| READBUFF= Restriction                                                 | 798 |
| Using DBSLICEPARM=                                                    | 799 |
| Using DBSLICE=                                                        | 799 |
| <i>Temporary Table Support for DB2 under z/OS</i>                     | 799 |
| <i>Calling Stored Procedures in DB2 under z/OS</i>                    | 799 |
| Overview                                                              | 799 |
| Examples                                                              | 800 |
| <i>ACCESS Procedure Specifics for DB2 under z/OS</i>                  | 802 |
| Key Information                                                       | 802 |
| ACCESS Procedure Examples                                             | 803 |
| <i>DBLOAD Procedure Specifics for DB2 under z/OS</i>                  | 804 |
| Key Information                                                       | 804 |
| DBLOAD Procedure Examples                                             | 805 |

|                                                                          |            |
|--------------------------------------------------------------------------|------------|
| <b>The DB2EXT Procedure .....</b>                                        | <b>806</b> |
| Overview .....                                                           | 806        |
| Syntax .....                                                             | 806        |
| PROC DB2EXT Statement Options .....                                      | 806        |
| FMT Statement .....                                                      | 807        |
| RENAME Statement .....                                                   | 807        |
| SELECT Statement .....                                                   | 807        |
| EXIT Statement .....                                                     | 808        |
| DB2EXT Procedure Examples .....                                          | 808        |
| <b>The DB2UTIL Procedure .....</b>                                       | <b>808</b> |
| Overview .....                                                           | 808        |
| DB2UTIL Syntax .....                                                     | 809        |
| PROC DB2UTIL Statement Options .....                                     | 810        |
| DB2UTIL Statements .....                                                 | 811        |
| Modifying DB2 Data .....                                                 | 812        |
| PROC DB2UTIL Example .....                                               | 813        |
| <b>Maximizing DB2 under z/OS Performance .....</b>                       | <b>814</b> |
| Assessing When to Tune Performance .....                                 | 814        |
| When the Resource Limit Facility Limits Execution Time .....             | 815        |
| Methods for Improving Performance .....                                  | 815        |
| Optimizing Your Connections .....                                        | 816        |
| <b>Passing SAS Functions to DB2 under z/OS .....</b>                     | <b>817</b> |
| <b>Passing Joins to DB2 under z/OS .....</b>                             | <b>819</b> |
| <b>SAS System Options, Settings, and Macros for DB2 under z/OS .....</b> | <b>819</b> |
| System Options .....                                                     | 819        |
| Values .....                                                             | 821        |
| Macros .....                                                             | 821        |
| <b>Bulk Loading for DB2 under z/OS .....</b>                             | <b>822</b> |
| Overview .....                                                           | 822        |
| Data Set Options for Bulk Loading .....                                  | 823        |
| File Allocation and Naming for Bulk Loading .....                        | 823        |
| Bulk Loading Examples .....                                              | 825        |
| <b>Locking in the DB2 under z/OS Interface .....</b>                     | <b>828</b> |
| <b>Naming Conventions for DB2 under z/OS .....</b>                       | <b>829</b> |
| Overview .....                                                           | 829        |
| Case Sensitivity and Special Characters in Object Names .....            | 829        |
| Using Table Aliases or Synonyms .....                                    | 830        |
| <b>Data Types for DB2 under z/OS .....</b>                               | <b>830</b> |
| Overview .....                                                           | 830        |
| Supported Data Types for DB2 under z/OS .....                            | 830        |
| DB2 Null and Default Values .....                                        | 831        |
| LIBNAME Statement Data Conversions .....                                 | 832        |
| ACCESS Procedure Data Conversions .....                                  | 834        |
| DBLOAD Procedure Data Conversions .....                                  | 835        |
| <b>Temporal Data for DB2 under z/OS .....</b>                            | <b>836</b> |
| Overview of Temporal Data for DB2 under z/OS .....                       | 836        |
| System Time and Bitemporal Data: History Tables .....                    | 836        |
| Data Set Options for Temporal Data .....                                 | 837        |
| <b>Understanding DB2 under z/OS Client/Server Authorization .....</b>    | <b>838</b> |

|                                                                        |            |
|------------------------------------------------------------------------|------------|
| Libref Connections .....                                               | 838        |
| Non-Libref Connections .....                                           | 840        |
| Known Issues with RRSASF Support .....                                 | 841        |
| <b>DB2 under z/OS Information for the Database Administrator .....</b> | <b>841</b> |
| How the Interface to DB2 Works .....                                   | 841        |
| How and When Connections Are Made .....                                | 842        |
| DDF Communication Database .....                                       | 842        |
| DB2 Attachment Facilities (CAF and RRSASF) .....                       | 843        |
| Accessing DB2 System Catalogs .....                                    | 844        |
| <b>Sample Programs for DB2 under z/OS Hosts .....</b>                  | <b>845</b> |

---

# System Requirements for SAS/ACCESS Interface to DB2 under z/OS

You can find information about system requirements for SAS/ACCESS Interface to DB2 under z/OS in the following locations:

- [System Requirements for SAS/ACCESS Interface to DB2 with SAS 9.4](#)
- [Third-Party Software Requirements for Use with SAS 9.4](#)

---

# Introduction to SAS/ACCESS Interface to DB2 under z/OS

For available SAS/ACCESS features, see [DB2 under z/OS supported features on page 100](#). For more information about DB2 under z/OS, see your DB2 under z/OS documentation.

---

**Note:** SAS/ACCESS Interface to DB2 under z/OS is not included with SAS Viya.

---

# LIBNAME Statement for the DB2 Engine under z/OS

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to DB2 under z/OS supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing DB2 under z/OS interface.

**LIBNAME** *libref db2 <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *db2*

specifies the SAS/ACCESS engine name for the DB2 under z/OS interface.

### *connection-options*

provides connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

### *USER=<'>DB2-user-name<'>*

lets you connect to a DB2 database with a user ID that is different from the default ID. The value for this option cannot exceed 8 characters. USER= is optional. If you specify USER=, you must also specify PASSWORD=. If USER= is omitted, your default user ID for your operating environment is used. If you do not specify the SCHEMA= or AUTHID= LIBNAME option, the value of the USER= option (if present) is used as the default schema.

Authentication options are not checked when the LIBNAME is issued. They are checked only when a statement involving the libref is run. However, for explicit pass-through, credentials are checked immediately.

**PASSWORD=<'>DB2-password<'>**

specifies the DB2 password that is associated with your DB2 user ID.  
**PASSWORD=** is optional. If you specify **USER=**, you must specify **PASSWORD=**.

**LOCATION=location**

maps to the location in the SYSIBM.LOCATIONS catalog in the communication database. In SAS/ACCESS Interface to DB2 under z/OS, the location is converted to the first level of a three-level table name:  
*location.authid.table*. DB2 Distributed Data Facility (DDF) Communication Database (CDB) makes the connection implicitly to the remote DB2 subsystem when DB2 receives a three-level name in an SQL statement.

**LOCATION=** is optional. If you omit it, SAS accesses the data from the local DB2 database unless you have specified a value for the **SERVER=** option. This option is not validated until you access a DB2 table. If you specify **LOCATION=**, you must also specify the [AUTHID=](#) option.

**SSID=DB2-subsystem-id**

specifies the DB2 subsystem ID to connect to at connection time. **SSID=** is optional. If you omit it, SAS connects to the DB2 subsystem that is specified in the **DB2SSID=** SAS system option. The DB2 subsystem ID is limited to four characters. For more information, see ["Values" on page 821](#).

**SERVER=DRDA-server**

specifies the DRDA server to which you want to connect. **SERVER=** lets you access DRDA resources stored at remote locations. Check with your system administrator for system names. You can connect to only one server per LIBNAME statement.

**SERVER=** is optional. If you omit it, you access tables from your local DB2 database unless you have specified a value for the [LOCATION= LIBNAME option](#).

Default: none.

| Task                                                                                                  | Information Resource                          |
|-------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| Accessing a database server on Linux, UNIX, or Windows using a libref                                 | <a href="#">REMOTE_DBTYPE= LIBNAME option</a> |
| Setting up DB2 z/OS so that SAS can connect to the DRDA server when the <b>SERVER=</b> option is used | Installation instructions for this interface  |
| Configuring SAS to use the <b>SERVER=</b> option                                                      |                                               |

**LIBNAME-options**

specifies how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to DB2 under z/OS, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 19.1** SAS/ACCESS LIBNAME Options

| Option               | Default Value   | Valid in CONNECT |
|----------------------|-----------------|------------------|
| ACCESS=              | none            |                  |
| ALLOWED_SQLCODES=    | none            |                  |
| AUTHDOMAIN=          | none            |                  |
| AUTHID=              | your user ID    |                  |
| CONNECTION=          | SHAREDREAD      | •                |
| CONNECTION_GROUP=    | none            | •                |
| DBCONINIT=           | none            | •                |
| DBCONTERM=           | none            | •                |
| DBCREATE_TABLE_OPTS= | none            |                  |
| DBGEN_NAME=          | DBMS            | •                |
| DBLIBINIT=           | none            |                  |
| DBLIBTERM=           | none            |                  |
| DBMSTEMP=            | NO              |                  |
| DBNULLKEYS=          | YES             |                  |
| DBSASLABEL=          | COMPAT          |                  |
| DBSLICEPARM=         | THREADED_APPS,2 |                  |
| DEFER=               | NO              | •                |
| DEGREE=              | ANY             |                  |
| DIRECT_EXE=          | none            |                  |
| DIRECT_SQL=          | YES             |                  |
| IN=                  | none            |                  |
| LOCATION=            | none            |                  |
| MULTI_DATASRC_OPT=   | NONE            |                  |

| Option                  | Default Value                           | Valid in CONNECT |
|-------------------------|-----------------------------------------|------------------|
| POST_STMT_OPTS=         | none                                    |                  |
| PRESERVE_COL_NAMES=     | NO                                      |                  |
| PRESERVE_TAB_NAMES=     | NO                                      |                  |
| READBUFF=               | 1                                       | •                |
| READ_ISOLATION_LEVEL=   | DB2 z/OS determines the isolation level |                  |
| READ_LOCK_TYPE=         | none                                    | •                |
| REMOTE_DBTYPE=          | ZOS                                     |                  |
| REREAD_EXPOSURE=        | NO                                      | •                |
| SCHEMA=                 | your user ID                            |                  |
| SPOOL=                  | YES                                     |                  |
| SQL_FUNCTIONS=          | none                                    |                  |
| SQL_FUNCTIONS_COPY=     | none                                    |                  |
| SQLGENERATION=          | none                                    |                  |
| UPDATE_ISOLATION_LEVEL= | DB2 z/OS determines the isolation level |                  |
| UPDATE_LOCK_TYPE=       | none                                    | •                |
| UTILCONN_TRANSIENT=     | YES                                     |                  |

## DB2 under z/OS LIBNAME Statement Example

In this example, the libref MYLIB uses the DB2 under z/OS interface to connect to the DB2 database that the SSID= option specifies, with a connection to the testserver remote server.

```
libname mylib db2 ssid=db2
      authid=myusr1 server=mysrv1;
proc print data=mylib.staff;
      where state='CA';
run;
```

# Data Set Options for DB2 under z/OS

All SAS/ACCESS data set options in this table are supported for SAS/ACCESS Interface to DB2 under z/OS. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 19.2** SAS/ACCESS Data Set Options for DB2 under z/OS

| Option           | Default Value                |
|------------------|------------------------------|
| AUTHID=          | current LIBNAME option value |
| BL_DBCURSOR=     | none                         |
| BL_DB2DATACLAS=  | none                         |
| BL_DB2DEVT_PERM= | SYSDA                        |
| BL_DB2DEVT_TEMP= | SYSDA                        |
| BL_DB2DISC=      | a generated data set name    |
| BL_DB2ERR=       | a generated data set name    |
| BL_DB2IN=        | a generated data set name    |
| BL_DB2LDCT1=     | none                         |
| BL_DB2LDCT2=     | none                         |
| BL_DB2LDCT3=     | none                         |
| BL_DB2LDEXT=     | GENRUN                       |
| BL_DB2MAP=       | a generated data set name    |
| BL_DB2MGMTCLAS=  | none                         |
| BL_DB2PRINT=     | a generated data set name    |
| BL_DB2PRNLOG=    | YES                          |
| BL_DB2REC=       | a generated data set name    |
| BL_DB2RECSP=     | 10                           |

| <b>Option</b>        | <b>Default Value</b>                                  |
|----------------------|-------------------------------------------------------|
| BL_DB2STRT=          | NO                                                    |
| BL_DB2SPC_PERM=      | 10                                                    |
| BL_DB2SPC_TEMP=      | 10                                                    |
| BL_DB2STORCLAS=      | none                                                  |
| BL_DB2TBLXST=        | NO                                                    |
| BL_DB2UNITCOUNT=     | none                                                  |
| BL_DB2UTID=          | user ID and second level DSN qualifier                |
| BULKLOAD=            | NO                                                    |
| BUSINESS_DATATYPE=   | TIMESTAMP(6)                                          |
| BUSINESS_TIMEFRAME=  | none                                                  |
| DBCOMMIT=            | current LIBNAME option value                          |
| DBCONDITION=         | none                                                  |
| DBCREATE_TABLE_OPTS= | current LIBNAME option value                          |
| DBFORCE=             | NO                                                    |
| DBGEN_NAME=          | DBMS                                                  |
| DBKEY=               | none                                                  |
| DBLABEL=             | NO                                                    |
| DBLARGETABLE=        | none                                                  |
| DBNULL=              | YES                                                   |
| DBNULLKEYS=          | current LIBNAME option value                          |
| DBSASLABEL=          | COMPAT                                                |
| DBSASTYPE=           | see “ <a href="#">Data Types for DB2 under z/OS</a> ” |
| DBSLICE=             | none                                                  |
| DBSLICEPARM=         | THREADED_APPS,2                                       |

| Option                  | Default Value                           |
|-------------------------|-----------------------------------------|
| DBTYPE=                 | none                                    |
| DEGREE=                 | ANY                                     |
| ERRLIMIT=               | 1                                       |
| IN=                     | current LIBNAME option value            |
| LOCATION=               | current LIBNAME option value            |
| NULCHAR=                | SAS                                     |
| NULCHARVAL=             | a blank character                       |
| OVERLAPS=               | <i>column names separated by commas</i> |
| POST_STMT_OPTS=         | none                                    |
| POST_TABLE_OPTS=        | none                                    |
| PRE_STMT_OPTS=          | none                                    |
| PRE_TABLE_OPTS=         | none                                    |
| PRESERVE_COL_NAMES=     | current LIBNAME option value            |
| READBUFF=               | current LIBNAME option value            |
| READ_ISOLATION_LEVEL=   | current LIBNAME option value            |
| READ_LOCK_TYPE=         | current LIBNAME option value            |
| SCHEMA=                 | current LIBNAME option value            |
| SYSTEM_TIMEFRAME=       | none                                    |
| TEMPORAL=               | none                                    |
| TRAP_151=               | NO                                      |
| UPDATE_ISOLATION_LEVEL= | current LIBNAME option value            |
| UPDATE_LOCK_TYPE=       | current LIBNAME option value            |

# SQL Pass-Through Facility Specifics for DB2 under z/OS

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the DB2 under z/OS interface:

- The *dbms-name* is DB2.
- The CONNECT statement is optional.
- The interface supports connections to multiple databases.
- The maximum length of an SQL statement is 1 megabyte.
- The LIBNAME options that are accepted in the CONNECT statement are indicated in [Table 19.1 on page 790](#).

## Examples

This example connects to DB2 and sends it two EXECUTE statements to process.

```
proc sql;
  connect to db2 (ssid=db2);
  execute (create view testid.whotookorders as
            select ordernum, takenby, firstname,
                   lastname, phone
            from testid.orders, testid.employees
            where testid.orders.takenby=
                  testid.employees.empid)
           by db2;
  execute (grant select on testid.whotookorders
            to myusr1) by db2;
  disconnect from db2;
quit;
```

This next example omits the optional CONNECT statement, uses the DB2SSID= value, and performs a query (shown in highlighting) on the Testid.Customers table.

```
proc sql;
  select * from connection to db2
    (select * from testid.customers where customer like '1%');
  disconnect from db2;
quit;
```

This example creates the Vlib.StockOrd SQL view that is based on the Testid.Orders table. Testid.Orders is an SQL/DS table that is accessed through DRDA.

```
libname vlib 'SAS-library'

proc sql;
  connect to db2 (server=testserver);
  create view vlib.stockord as
    select * from connection to db2
      (select ordernum, stocknum, shipto, dateorderd
       from testid.orders);
  disconnect from db2;
quit;
```

## Special Catalog Queries

SAS/ACCESS Interface to DB2 under z/OS supports the following special queries. You can use the queries to call functions in ODBC-style function application programming interfaces (APIs). Here is the general format of the special queries:

`DBMS::SQLAPI('parameter-1', 'parameter-n')`

`DBMS::`

is required to distinguish special queries from regular queries. `DBMS::` is not case sensitive.

`SQLAPI`

is the specific API that is being called. an SQLAPI name is not case sensitive.

`'parameter n'`

is a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, this call to PrimaryKeys matches table names such as myatest and my\_test:

```
select * from connection to db2
  (DBMS::PrimaryKeys( '', 'JillSmith', 'my_test') );
```

Use the escape character to search only for the my\_test table:

```
select * from connection to db2
  (DBMS::PrimaryKeys( '', 'JillSmith', 'my\_test') );
```

SAS/ACCESS Interface to DB2 under z/OS supports these special queries.

`DBMS::ForeignKeys ('PK-Catalog', 'PK-Schema', 'PK-TableName', 'FK-Catalog', 'FK-Schema', 'FK-TableName'`

returns a list of all columns that comprise foreign keys that match the specified arguments. If no arguments are specified, all accessible foreign key columns and information are returned.

`DBMS::Indexes ('Catalog', 'TableOwner', 'TableName')`

returns the indexes that are specified in the catalog for a given user and table.

DBMS::PrimaryKeys ('Catalog', 'TableOwner', 'TableName')  
returns a list of all columns that compose the primary key that matches the specified table. A primary key can be composed of one or more columns. If you do not specify any table name, then this special query fails.

DBMS::RowCount ('Catalog', 'TableOwner', 'TableName')  
returns the number of rows for a specified table if statistics have been gathered for that table. Otherwise, this query returns a value of -1. The query also returns a value of -1 if the row describes a view, alias, accelerator-only table, or a temporary table.

---

# Autopartitioning Scheme for DB2 under z/OS

---

## Overview

Autopartitioning for SAS/ACCESS Interface to DB2 under z/OS is a modulo (MOD) method. Threaded Reads for DB2 under z/OS involve a trade-off. A threaded Read with even distribution of rows across the threads substantially reduces elapsed time for your SAS step. So your job completes in less time. This is positive for job turnaround time, particularly if your job needs to complete within a constrained period of time. However, threaded Reads always increase the CPU time of your SAS job and the workload on DB2. If increasing CPU consumption or increasing DB2 workload for your job are unacceptable, you can turn threaded Reads off by specifying DBSLICEPARM=NONE. To turn off threaded Reads for all SAS jobs, specify DBSLICEPARM=NONE in the SAS restricted options table.

For general information about this feature, see [“Autopartitioning Techniques in SAS/ACCESS” on page 76](#).

---

## Autopartitioning Restrictions

SAS/ACCESS Interface to DB2 under z/OS places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here are the column types that you can partition.

- INTEGER
- SMALLINT
- DECIMAL

You must confine eligible DECIMAL columns to an integer range—specifically, DECIMAL columns with precision that is less than 10. For example, DECIMAL(5,0) and DECIMAL(9,2) are eligible.

## Column Selection for MOD Partitioning

If multiple columns are eligible for partitioning, the engine queries the DB2 system tables for information about identity columns and simple indexes. Based on the information about the identity columns, simple indexes, column types, and column nullability, the partitioning column is selected in order by priority.

- 1 Identity column
- 2 Unique simple index: SHORT or INT, integral DECIMAL, and then nonintegral DECIMAL
- 3 Nonunique simple index: SHORT or INT (NOT NULL), integral DECIMAL (NOT NULL), and then nonintegral DECIMAL (NOT NULL)
- 4 Nonunique simple index: SHORT or INT (nullable), integral DECIMAL (nullable), and then nonintegral DECIMAL (nullable)
- 5 SHORT or INT (NOT NULL), integral DECIMAL (NOT NULL), and then nonintegral DECIMAL (NOT NULL)
- 6 SHORT or INT (nullable), integral DECIMAL (nullable), and then nonintegral DECIMAL (nullable)

If a nullable column is selected for autopartitioning, the SQL statement `OR<column-name>IS NULL` is appended at the end of the SQL code that is generated for one read thread. This ensures that any possible NULL values are returned in the result set.

## How WHERE Clauses Restrict Autopartitioning

Autopartitioning does not select a column to be the partitioning column if it appears in a SAS WHERE clause. For example, the following DATA step cannot use a threaded Read to retrieve the data because all numeric columns in the table (see the table definition in “[Using DBSLICE=](#) on page 799) are in the WHERE clause.

```
data work.locemp;
  set trlib.MYEMPS;
  where EMPNUM<=30 and ISTENURE=0 and
        SALARY<=35000 and NUMCLASS>2;
run;
```

## READBUFF= Restriction

If Autopartitioning is specified or being done by default and if `READBUFF=` is set to a value higher than 1, then a program fails if there are no columns eligible for partitioning. Only numeric columns are used for partitioning.

If no eligible columns are available for partitioning, then the following error message is sent to the SAS log, and the program ends abnormally.

ERROR: (ACCDB2M240E) Couldn't fallback to non-sliced with READBUFF>1.

To fix this, ensure that an eligible column is available for partitioning. Alternatively, specify READBUFF=1 or set **DBSLICEPARM=NONE**.

---

## Using DBSLICEPARM=

SAS/ACCESS Interface to DB2 under z/OS defaults to two threads when you use autopartitioning.

---

## Using DBSLICE=

To achieve the best possible performance when using threaded Reads, specify the **DBSLICE=** data set option for DB2 in your SAS operation.

---

## Temporary Table Support for DB2 under z/OS

SAS/ACCESS Interface to DB2 under z/OS supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

---

## Calling Stored Procedures in DB2 under z/OS

---

### Overview

A stored procedure is one or more SQL statements or supported third-generation languages (3GLs, such as C) statements that are compiled into a single procedure that exists in DB2. Stored procedures might contain static (hardcoded) SQL statements. Static SQL is optimized better for some DBMS operations. In a carefully managed DBMS environment, programmers and database administrators can know the exact SQL to execute.

SAS usually generates SQL dynamically. However, the database administrator can encode static SQL in a stored procedure and therefore restrict SAS users to a tightly controlled interface. When you use a stored procedure call, you must specify a schema.

SAS/ACCESS support for stored procedure includes passing input parameters, retrieving output parameters into SAS macro variables, and retrieving the result set into a SAS table. Although DB2 stored procedures can return multiple result sets, SAS/ACCESS Interface to DB2 under z/OS can retrieve only a single result set.

You can call stored procedures only from PROC SQL.

## Examples

### Example 1: Specify a Basic Call

Use CALL statement syntax to call a stored procedure.

```
call "schema".stored_proc
```

The simplest way to call a stored procedure is to use the EXECUTE statement in PROC SQL. In this example, you execute STORED\_PROC by using a CALL statement. SAS does not capture the result set.

```
proc sql;
  connect to db2;
  execute (call "schema".stored_proc);
  quit;
```

### Example 2: Specify One Input Parameter That Returns a Result Set

You can also return the result set to a SAS table. In this example, STORED\_PROC is executed using a CALL statement. The result is returned to a SAS table, SasResults.

```
proc sql;
  connect to db2;
  create table sasresults as select * from connection
    to db2 (call "schema".stored_proc);
  quit;
```

---

## Example 3: Specify Three Output Parameters

The CALL statement syntax supports passing of parameters. You can specify such input parameters as numeric constants, character constants, or a null value. You can also pass input parameters by using SAS macro variable references. To capture the value of an output parameter, a SAS macro variable reference is required. This example uses a constant (1), an input/output parameter (:INOUT), and an output parameters (:OUT). Not only is the result set returned to the SAS results table, the SAS macro variables INOUT and OUT capture the parameter outputs.

```
proc sql;
connect to db2;
%let INOUT=2;
create table sasresults as select * from connection to db2
  (call "schema".stored_proc (1,:INOUT,:OUT));
quit;
```

---

## Example 4: Pass a NULL Parameter

In these calls, NULL is passed as the parameter to the DB2 stored procedure.

- null string literals in the call
 

```
call proc(' ');
call proc("")
```
- literal period or literal NULL in the call
 

```
call proc(.)
call proc(NULL)
```
- SAS macro variable set to NULL string
 

```
%let charparm=;
call proc(:charparm)
```
- SAS macro variable set to period (SAS numeric value is missing)
 

```
%let numparm=.;
call proc(:numparm)
```

Only the literal period and the literal NULL work generically for both DB2 character parameters and DB2 numeric parameters. For example, a DB2 numeric parameter would reject "" and %let numparm=.; would not pass a DB2 NULL for a DB2 character parameter. As a literal, a period passes NULL for both numeric and character parameters. However, when it is in a SAS macro variable, it constitutes a NULL only for a DB2 numeric parameter.

You cannot pass NULL parameters by omitting the argument. For example, you cannot use this call to pass three NULL parameters.

```
call proc(,,)
```

You could use this call instead.

```
call proc(NULL,NULL,NULL)
```

---

## Example 5: Specify a Schema

Use standard CALL statement syntax to execute a stored procedure that exists in another schema, as shown in this example.

```
proc sql;
  connect to db2;
  execute (call otherschema.stored_proc);
  quit;
```

If the schema is in mixed case or lowercase, enclose the schema name in double quotation marks.

```
proc sql;
  connect to db2;
  execute (call "lowschema".stored_proc);
  quit;
```

---

## Example 6: Execute Remote Stored Procedures

If the stored procedure exists on a different DB2 instance, specify it with a valid three-part name.

```
proc sql;
  connect to db2;
  create table sasresults as select * from connection to db2
    (call otherdb2.procschema.prod5 (1, NULL));
  quit;
```

---

# ACCESS Procedure Specifics for DB2 under z/OS

---

## Key Information

For general information about this feature, see the [ACCESS procedure on page 1421](#).

SAS/ACCESS Interface to DB2 under z/OS supports all [ACCESS procedure statements](#) in interactive line, noninteractive, and batch modes.

**Note:** SAS still supports this legacy procedure. However, to access your DBMS data more directly the best practice is to use the LIBNAME statement for DB2 under z/OS or the SQL pass-through facility. For more information, see “[LIBNAME Statement for the DB2 Engine under z/OS](#)” on page 788 and “[SQL Pass-Through Facility Specifics for DB2 under z/OS](#)” on page 795.

Here are the ACCESS procedure specifics for the DB2 under z/OS interface.

- The DBMS= value is db2.
- The *database-connection-arguments* are SSID=, SERVER=, and LOCATION=. These options function in the same way as the corresponding options in the LIBNAME statement.
- Here is the TABLE= statement:

TABLE= <*authorization-id*.>*table-name*

identifies the DB2 table or DB2 view that you want to use to create an access descriptor. The *table-name* is limited to 18 characters. The TABLE= statement is required.

The *authorization-id* is a user ID or group ID that is associated with the DB2 table. The authorization ID is limited to eight characters. If you omit the authorization ID, DB2 uses your TSO (or z/OS) user ID. In batch mode, however, you must specify an authorization ID. Otherwise, an error message is generated.

## ACCESS Procedure Examples

This example creates an access descriptor and a view descriptor that are based on DB2 data.

```
options linesize=80;
libname adlib 'SAS-library';
libname vlib 'SAS-library';

proc access dbms=db2;

/* create access descriptor */
create adlib.customr.access;
table=testid.customers;
ssid=db2;
assign=yes;
rename customer=custnum;
format firstorder date7. ;
list all;

/* create vlib.usacust view */
create vlib.usacust.view;
select customer state zipcode name
      firstorder;
subset where customer like '1%';
run;
```

This next example uses the SERVER= statement to access the SQL/DS table Testid.Orders from a remote location. Access and view descriptors are then created based on the table.

```
libname adlib 'SAS-library';
libname vlib 'SAS-library';

proc access dbms=db2;
  create adlib.customr.access;
  table=testid.orders;
  server=testserver;
  assign=yes;
  list all;

  create vlib.allord.view;
  select ordernum stocknum shipto dateorderd;

  subset where stocknum = 1279;
run;
```

## DBLOAD Procedure Specifics for DB2 under z/OS

### Key Information

For general information about this feature, see the [DBLOAD procedure on page 1455](#).

SAS/ACCESS Interface to DB2 under z/OS supports all [DBLOAD procedure statements](#) in interactive line, noninteractive, and batch modes.

**Note:** SAS still supports this legacy procedure. However, to send data from SAS to your DBMS more directly the best practice is to use the LIBNAME statement for DB2 under z/OS or the SQL pass-through facility. For more information, see [“LIBNAME Statement for the DB2 Engine under z/OS” on page 788](#) and [“SQL Pass-Through Facility Specifics for DB2 under z/OS” on page 795](#).

Here are the DBLOAD procedure specifics for SAS/ACCESS Interface to DB2 under z/OS.

- The DBMS= value is DB2.
- The *database-connection-arguments* are SSID= and SERVER=.
- Here is the NULLS= statement:

NULLS *variable-identifier-1* =Y|N|D < . . . *variable-identifier-n* =Y|N|D >  
lets you specify whether DB2 columns that are associated with the listed SAS variables allow NULL values. By default, all columns accept NULL values.

The NULLS statement accepts any one of these values.

- Y: Specifies that the column accepts NULL values. This is the default.
  - N: Specifies that the column does not accept NULL values.
  - D: Specifies that the column is specified as NOT NULL WITH DEFAULT.
- For information about NULL values that is specific to DB2, see “[DB2 Null and Default Values](#)” on page 831.

- Here is the TABLE= statement:

**TABLE= <authorization-id.>table-name;**

identifies the DB2 table or DB2 view that you want to use to create an access descriptor. The *table-name* is limited to 18 characters. The TABLE= statement is required.

The *authorization-id* is a user ID or group ID that is associated with the DB2 table. The authorization ID is limited to eight characters. If you omit the authorization ID, DB2 uses your TSO (or z/OS) user ID. However, in batch mode, you must specify an authorization ID or an error message is generated.

## DBLOAD Procedure Examples

This example creates a new DB2 table, Testid.Invoice, from the Dlib.Invoice data file. The AmtBilled column and the fifth column in the table (AmountInUS) are renamed. You must have the appropriate privileges before you can create new DB2 tables.

```
libname adlib 'SAS-library';
libname dlib 'SAS-library';

proc dbload dbms=db2 data=dlib.invoice;
  ssid=db2;
  table=testid.invoice;
  accdesc=adlib.invoice;
  rename amtbilled=amountbilled
            5=amountindollars;
  nulls invoicenum=n amtbilled=n;
  load;
run;
```

For example, you can create a SAS data set, Work.Schedule, that includes the names and work hours of your employees. You can use the SERVER= command to create the DB2 table, Testid.Schedule, and load it with the schedule data on the DRDA resource, TestServer, as shown in this example.

```
libname adlib 'SAS-library';

proc dbload dbms=db2 data=work.schedule;
  in sample;
  server=testserver;
  accdesc=adlib.schedule;
  table=testid.schedule;
  list all;
  load;
```

```
run;
```

---

# The DB2EXT Procedure

---

## Overview

The DB2EXT procedure creates SAS data sets from DB2 under z/OS data. PROC DB2EXT runs interactively, noninteractively, and in batch mode. The generated data sets are not password protected. However, you can edit the saved code to add password protection.

PROC DB2EXT ensures that all SAS names that are generated from DB2 column values are unique. A numeric value is appended to the end of a duplicate name. If necessary, the procedure truncates the name when appending the numeric value.

---

**Note:** Support for the DB2EXT procedure provides compatibility with the SAS 5 version of SAS/ACCESS Interface to DB2 under z/OS. It is not added to other SAS/ACCESS DBMS interfaces, and enhancements of this procedure for future releases of SAS/ACCESS are not guaranteed. It is recommended that you write new applications by using LIBNAME features.

---

## Syntax

Here is the syntax for the DB2EXT procedure.

```
PROC DB2EXT <options>;
  FMT column-number-1='SAS-format-name-1'
    <... column-number-n='SAS-format-name-n'>;
  RENAME column-number-1='SAS-name-1'
    <... column-number-n='SAS-name-n'>;
  SELECT DB2-SQL-statement;
  EXIT;
```

---

## PROC DB2EXT Statement Options

### IN=SAS-data-set

specifies a mapping data set that contains information such as DB2 names, SAS variable names, and formats for input to PROC DB2EXT. This option is available for use only with previously created mapping data sets. You cannot create new mapping data sets with DB2EXT.

**OUT=SAS-data-set | libref.SAS-data-set**

specifies the name of the SAS data set that is created. If you omit OUT=, the data set is named "work.DATAn", where n is a number that is sequentially updated. The data set is not saved when your SAS session ends. If a file with the name that you specify in the OUT= option already exists, it is overwritten. However, you receive a warning that this is going to happen.

**SSID=subsystem-name**

specifies the name of the DB2 subsystem that you want to access. If you omit SSID=, the subsystem name defaults to DB2. The subsystem name defaults to the subsystem that is specified in the DB2SSID= option. It defaults to DB2 only if neither the SSID= option nor the DB2SSID= option are specified.

## FMT Statement

**FMT column-number-1='SAS-format-name-1'  
<... column-number-n='SAS-format-name-n'>;**

The FMT statement assigns a SAS output format to the DB2 column that is specified by *column-number*. The *column-number* is determined by the order in which you list the columns in your SELECT statement. If you use SELECT \*, the *column-number* is determined by the order of the columns in the database. You must enclose the format name in single quotation marks. You can specify multiple column formats in a single FMT statement.

## RENAME Statement

**RENAME column-number-1='SAS-name-1'  
<... column-number-n='SAS-name-n'>;**

The RENAME statement assigns the *SAS-name* to the DB2 column that is specified by *column-number*. The *column-number* is determined by the order in which you list the columns in your SELECT statement. If you use SELECT \*, the *column-number* is determined by the order of the columns in the database.

You can rename multiple columns in a single RENAME statement.

## SELECT Statement

**SELECT DB2-SQL-statement;**

The *DB2-SQL-statement* specifies the DB2 data that you want to include in the SAS data set. You can specify table names, column names, and data subsets in your SELECT statement. For example, this statement selects all columns from the Employee table and includes only employees whose salary is greater than \$40,000.

```
select * from employee where salary > 40000;
```

---

## EXIT Statement

**EXIT;**

The EXIT statement terminates the procedure without further processing.

---

## DB2EXT Procedure Examples

This code creates a SAS data set named MyLib.NoFmt that includes three columns from the DB2 table EmplInfo. The RENAME statement changes the name of the third column that is listed in the SELECT statement (from `firstname` in the DB2 table to `fname` in the SAS data set).

```
/* specify the SAS library where the SAS data set is to be saved */

libname mylib 'userid.xxx';

proc db2ext ssid=db25 out=mylib.nofmt;
    select employee, lastname, firstname from sasdemo.emplinfo;
    rename 3=fname;
run;
```

This code uses a mapping file to specify which data to include in the SAS data set and how to format that data.

```
/* specify the SAS library where the SAS data set is to be saved */
libname mylib 'userid.xxx';

/* specify the SAS library that contains the mapping data set */
libname inlib 'userid.maps';

proc db2ext in=inlib.mapping out=mylib.mapout ssid=db25;
run;
```

---

## The DB2UTIL Procedure

---

### Overview

You can use the DB2UTIL procedure to insert, update, or delete rows in a DB2 table using data from a SAS data set. You can choose one of two methods of processing: creating an SQL output file or executing directly. PROC DB2UTIL runs interactively, noninteractively, or in batch mode.

---

**Note:** Support for the DB2UTIL procedure provides compatibility with SAS 5 version of SAS/ACCESS Interface to DB2 under z/OS. It is not added to other SAS/ACCESS DBMS interfaces, and enhancements of this procedure for future releases of SAS/ACCESS are not guaranteed. It is recommended that you write new applications by using LIBNAME features.

---

The DB2UTIL procedure uses the data in an input SAS data set, along with your mapping specifications, to generate SQL statements that modify the DB2 table. The DB2UTIL procedure can perform these functions.

**DELETE**

deletes rows from the DB2 table according to the search condition that you specify.

**INSERT**

builds rows for the DB2 table from the SAS observations, according to the map that you specify and inserts the rows.

**UPDATE**

specifies new column values in your DB2 table by using the SAS variable values that are indicated in your map.

When you execute the DB2UTIL procedure, you specify an input SAS data set, an output DB2 table, and how to modify the data. To generate data, you must also provide instructions for mapping the input SAS variable values to the appropriate DB2 columns.

In each execution, the procedure can generate and execute SQL statements to perform one type of modification only. However, you can also provide your own SQL statements (except the SQL SELECT statement) to perform various modifications against your DB2 tables, and the procedure executes them.

For more information about the types of modifications that are available and how to use them, see “[Modifying DB2 Data](#)” on page 812. For an example of how to use this procedure, see the [PROC DB2UTIL example](#) on page 813.

---

## DB2UTIL Syntax

The PROC DB2UTIL statement calls the [DB2UTIL procedure](#) on page 808. These statements are used with PROC DB2UTIL.

**PROC DB2UTIL** <*options*>;

**MAPTO** SAS-name-1=DB2-name-1 <...SAS-name-n=DB2-name-n>;

**RESET** ALL|SAS-name| COLS;

**SQL** SQL-statement;

**UPDATE**:

**WHERE** SQL-WHERE-clause;

**ERRLIMIT**=error-limit;

**EXIT**;

## PROC DB2UTIL Statement Options

**DATA=SAS-data-set | <libref.>SAS-data-set**

specifies the name of the SAS data set that contains the data with which you want to update the DB2 table. DATA= is required unless you specify an SQL file with the SQLIN= option.

**TABLE=DB2-tablename**

specifies the name of the DB2 table that you want to update. TABLE= is required unless you specify an SQL file with the SQLIN= option.

**FUNCTION= D | I | U | DELETE | INSERT | UPDATE**

specifies the type of modification to perform on the DB2 table by using the SAS data set as input. For a detailed description of this option, see “[Modifying DB2 Data](#)” on page 812. FUNCTION= is required unless you specify an SQL file with the SQLIN= option.

**COMMIT=number**

specifies the maximum number of SQL statements to execute before issuing an SQL COMMIT statement to establish a synchpoint. The default is 3.

**ERROR=fileref | fileref.member**

specifies an external file where error information is logged. When DB2 issues an error return code, the procedure writes all relevant information, including the SQL statement that is involved, to this external file. If you omit the ERROR= statement, the procedure writes the error information to the SAS log.

**LIMIT=number**

specifies the maximum number of SQL statements to issue in an execution of the procedure. The default value is 5000. If you specify LIMIT=0, no limit is specified. The procedure processes the entire data set regardless of its size.

**SQLIN=fileref | fileref.member**

specifies an intermediate SQL output file that is created by a prior execution of PROC DB2UTIL by using the SQLOUT= option. The file that is specified by SQLIN= contains SQL statements to update a DB2 table. If you specify an SQLIN= file, the procedure reads the SQL statements and executes them in line mode. When you specify an SQLIN= file, DATA=, TABLE=, and SQLOUT= are ignored.

**SQLOUT=fileref | fileref.member**

specifies an external file where the generated SQL statements are to be written. This file is either a z/OS sequential data set or a member of a z/OS partitioned data set. Use this option to update or delete data. When you specify the SQLOUT= option, the procedure edits your specifications, generates the SQL statements to perform the update, and writes them to the external file for later execution. When they are used as input in the later execution, the procedure passes them to DB2.

**SSID=subsystem-name**

specifies the name of the DB2 subsystem that you want to access. If you omit DB2SSID=, the subsystem name defaults to DB2. For more information, see “[Values](#)” on page 821.

---

## DB2UTIL Statements

---

### MAPTO Statement

MAPTO SAS-name-1=DB2-name-1<... SAS-name-n=DB2-name-n>;

The MAPTO statement maps the SAS variable name to the DB2 column name. You can specify as many values in one MAPTO statement as you want.

---

### RESET Statement

RESET ALL | SAS-name | COLS;

Use the RESET statement to erase the editing that was done to SAS variables or DB2 columns. The RESET statement can perform one or more of these actions:

ALL

resets all previously entered map and column names to default values for the procedure.

SAS-name

resets the map entry for that SAS variable.

COLS

resets the altered column values.

---

### SQL Statement

SQL SQL-statement;

The SQL statement specifies an SQL statement that you want the procedure to execute dynamically. The procedure rejects SQL SELECT statements.

---

### UPDATE Statement

UPDATE;

The UPDATE statement causes the table to be updated by using the mapping specifications that you provide. If you do not specify an input or output mapping data set or an SQL output file, the table is updated by default.

If you have specified an output mapping data set in the SQLOUT= option, PROC DB2UTIL creates the mapping data set and ends the procedure. However, if you specify UPDATE, the procedure creates the mapping data set and updates the DB2 table.

---

## WHERE Statement

WHERE SQL-WHERE-clause;

The WHERE statement specifies the SQL WHERE clause that you want to use to update the DB2 table. This statement is combined with the SQL statement generated from your mapping specifications. Any SAS variable names in the WHERE clause are substituted at that time, as shown in this example.

```
where db2col = %sasvar;
```

---

## ERRLIMIT Statement

ERRLIMIT=error-limit;

The ERRLIMIT statement specifies the number of DB2 errors that are permitted before the procedure terminates.

---

## EXIT Statement

EXIT;

The EXIT statement exits from the procedure without further processing. No output data is written, and no SQL statements are issued.

---

# Modifying DB2 Data

---

## Overview

The DB2UTIL procedure generates SQL statements by using data from an input SAS data set. However, the SAS data set plays a different role for each type of modification that is available through PROC DB2UTIL. These sections show how you use each type and how each type uses the SAS data set to make a change in the DB2 table.

---

## Inserting Data

You can insert observations from a SAS data set into a DB2 table as rows in the table. To use this insert function, name the SAS data set and the DB2 table in the PROC DB2UTIL statement. You can then use the MAPTO statement to map values

from SAS variables to columns in the DB2 table. If you do not want to insert the values for all variables in the SAS data set into the DB2 table, map only the variables that you want to insert. However, you must map all DB2 columns to a SAS column.

---

## Updating Data

You can change the values in DB2 table columns by replacing them with values from a SAS data set. You can change a column value to another value for every row in the table, or you can change column values only when certain criteria are met. For example, you can change the value of the NUM DB2 column to 10 for every row in the table. You can also change the value of the NUM DB2 column to the value in the NUMBER SAS variable if the DB2 column name value and the SAS data set variable name match.

You specify the name of the SAS data set and the DB2 table to be updated when you execute PROC DB2UTIL. You can specify that only certain variables be updated by naming only those variables in your mapping specifications.

You can use the WHERE clause to specify that only the rows on the DB2 table that meet certain criteria are updated. For example, you can use the WHERE clause to specify that only the rows with a certain range of values are updated. Or you can specify that rows to be updated when a certain column value in the row matches a certain SAS variable value in the SAS data set. In this case, you could have a SAS data set with several observations in it. For each observation in the data set, the DB2UTIL procedure updates the values for all rows in the DB2 table that have a matching value. The procedure then goes on to the next observation in the SAS data set and continues to update values in DB2 columns in rows that meet the comparison criteria.

---

## Deleting Data

You can remove rows from a DB2 table when a certain condition is met. You can delete rows from the table when a DB2 column value in the table matches a SAS variable value in the SAS data set. Name the DB2 table from which you want to delete rows and the SAS data set that contains the target deletion values in the PROC DB2UTIL statement. You can then use the WHERE statement to specify the DB2 column name and the SAS variable whose values must match before the deletion is performed. To delete values that are based on criteria other than values in SAS data variables, you can use an SQL DELETE statement. For example, you can delete every row with a department number of 600.

---

## PROC DB2UTIL Example

This example uses the UPDATE function in PROC DB2UTIL to update a list of telephone extensions from a SAS data set. The primary list of extensions is in the DB2 table Testid.Employees and is updated from the SAS data set Trans. First, create the SAS data set.

```

options sastrace=',,d';

data trans;
  empno=321783;
  ext='3999';
  output;
  empno=320001;
  ext='4321';
  output;
  empno=212916;
  ext='1300';
  output;
run;

```

Next, specify the data set in PROC DB2UTIL.

```

proc db2util data=trans table=testid.employees function=u;
  mapto ext=phone;
  where empid=%empno;
  update;
run;

```

The row that includes EMPID=320001 is not found in the Testid.Employees table and is therefore not updated. You can ignore the warning in the SAS log.

## Maximizing DB2 under z/OS Performance

### Assessing When to Tune Performance

Among the factors that affect DB2 performance are the size of the table that is being accessed and the form of the SQL SELECT statement. If the table that is being accessed is larger than 10,000 rows (or 1,000 pages), you should evaluate all SAS programs that access the table directly. When you evaluate the programs, consider these questions.

- Does the program need all columns that the SELECT statement retrieves?
- Do the WHERE clause criteria retrieve only those rows that are needed for subsequent analysis?
- Is the data going to be used by more than one procedure in one SAS session? If so, consider extracting the data into a SAS data set for SAS procedures to use instead of allowing the data to be accessed directly by each procedure.
- Do the rows need to be in a particular order? If so, can an indexed column be used to order them? If there is no index column, is DB2 doing the sort?
- Do the WHERE clause criteria allow DB2 to use the available indexes efficiently?
- What type of locks does DB2 need to acquire?

- Are the joins being passed to DB2?
- Can your DB2 system use parallel processing to access the data more quickly?

## When the Resource Limit Facility Limits Execution Time

The DB2 Resource Limit Facility limits execution time of dynamic SQL statements. If the time limit is exceeded, the dynamic statement is terminated and the SQL code -905 is returned. This list describes several situations in which the RLF could stop a user from consuming large quantities of CPU time.

- an extensive join of DB2 tables with the SAS SQL procedure
- an extensive search by the FSEDIT, FSVIEW, or FSBROWSE procedures or an SCL application
- any extensive extraction of data from DB2
- an extensive select
- an extensive load into a DB2 table (In this case, you can break up the load by lowering the commit frequency, or you can use the [bulk-load facility on page 822](#) through SAS/ACCESS Interface to DB2 under z/OS.)

## Methods for Improving Performance

You can do several things in your SAS application to improve DB2 engine performance.

- Specify the SAS system option `SASTRACE=' , , , d'`. This option prints to the SAS log the dynamic SQL that the DB2 engine generated and all other SQL that the DB2 engine executed. You can then verify that all WHERE clauses, PROC SQL joins, and ORDER BY clauses are being passed to DB2. This option is for debugging purposes and should not be specified once the SAS application is used in production. Specify `SASTRACE=OFF` to disable this option.
- Verify that all SAS procedures and DATA steps that read DB2 data share connections where possible. You can do this by using one libref to reference all SAS applications that read DB2 data and by accepting the default value of `SHAREDREAD` for the `CONNECTION=` option.
- If your DB2 subsystem supports parallel processing, you can assign a value to the CURRENT DEGREE special register. Setting this register might enable your SQL query to use parallel operations. You can specify the special register by using the LIBNAME options `DBCONINIT=` or `DBLIBINIT=` with the `SET` statement as shown in this example:

```
libname mydb2 db2 dbconinit="SET CURRENT DEGREE='ANY'";
```

- Use the view descriptor WHERE clause or the `DBCONDITION=` option to pass WHERE clauses to DB2. You can also use these methods to pass sort operations to DB2 with the ORDER BY clause instead of performing a sort within SAS.

- If you are using a SAS application or an SCL application that reads the DB2 data twice, let the DB2 engine spool the DB2 data. This happens by default because the default value for the SPOOL= option is YES.

The spool file is read both when the application rereads the DB2 data and when the application scrolls forward or backward through the data. If you do not use spooling but need to scroll backward through the DB2 table, the DB2 engine must start reading from the beginning of the data to the row that you want to access.

- Use the SQL procedure to pass joins to DB2 instead of using the MATCH MERGE capability (that is, merging with a BY statement) of the DATA step.
- Use the DBKEY= option when you are doing SAS processing that involves the KEY= option. When you use the DBKEY= option, the DB2 engine generates a WHERE clause that uses parameter markers. During the execution of the application, the values for the key are substituted into the parameter markers in the WHERE clause.

If you do not use the DBKEY= option, the entire table is retrieved into SAS, and the join is performed in SAS.

- Consider using stored procedures when they can improve performance in client/server applications by reducing network traffic. You can execute a stored procedure by using the DBCONINIT= or DBLIBINIT= LIBNAME options.
- Use the [READBUFF= LIBNAME](#) option to retrieve records in blocks instead of one at a time.

## Optimizing Your Connections

The DB2 engine supports more than one connection to DB2 per SAS session. This enables you to separate tasks that fetch rows from a cursor from tasks that must issue commits. This separation eliminates having to resynchronize the cursor, prepare the statement, and fetch rows until you are positioned back on the row that you were on. This separation enables commit tasks to avoid locking contention and act sooner because the tasks do not have to wait for cursors to close before resynchronizing. In general, tables that are opened for input fetch from cursors do not issue commits. Tables that are opened for update might issue commits, and tables that are opened for output do issue commits.

You can control how the DB2 engine uses connections by using the [CONNECTION=](#) option in the LIBNAME statement. At one extreme is CONNECTION=UNIQUE, which causes each table access, whether it is for input, update, or output, to create and use its own connection. Conversely, CONNECTION=SHARED means that only one connection is made, and that input, update, and output accesses all share that connection.

The default value for the CONNECTION= option is CONNECTION=SHAREDREAD, which means that tables opened for input share one connection. Update and output openings obtain their own connections. CONNECTION=SHAREDREAD allows for the best separation between tasks that fetch from cursors and tasks that must issue commits, eliminating the resynchronizing of cursors.

The values GLOBAL and GLOBALREAD perform similarly to SHARED and SHAREDREAD. The difference is that you can share the given connection across any of the librefs that you specify as GLOBAL or GLOBALREAD.

Although the default value of CONNECTION=SHAREDREAD is usually optimal, at times another value might be better. If you must use multiple librefs, you might want to specify them each as GLOBALREAD. In this case, you have one connection for all of your input openings, regardless of which libref you use, as opposed to one connection per libref for input openings. In a single-user environment (as opposed to a server session), you might know that you do not have multiple openings occurring at the same time. In this case, you might want to use SHARED or GLOBAL for multiple librefs. By using such a value, you eliminate the overhead of creating separate connections for input, update, and output transactions. If you have only one opening at a time, you eliminate the problem of resynchronizing input cursors if a commit occurs.

Another reason for using SHARED or GLOBAL is the case of opening a table for output while opening another table within the same database for input. This can result in a -911 deadlock situation unless both opens occur in the same connection.

As explained in “[DB2 under z/OS Information for the Database Administrator](#)” on [page 841](#), the first connection to DB2 is made from the main SAS task. Subsequent connections are made from corresponding subtasks, which the DB2 engine attaches; DB2 allows only one connection per task. Due to the system overhead of intertask communication, the connection established from the main SAS task is a faster connection in terms of CPU time. Because this is true, you can expect better performance (less CPU time) if you use the first connection for these operations when you read or write large numbers of rows. If you read-only rows, SHAREDREAD or GLOBALREAD can share the first connection. However, if you are both reading and writing rows (input and output opens), you can use CONNECTION=UNIQUE to make each opening use the first connection. UNIQUE causes each opening to have its own connection. Suppose you have only one opening at a time and some are input while others are output for large amounts of data. The performance benefit of using the main SAS task connection far outweighs the overhead of establishing a new connection for each opening.

The utility connection is another type of connection that the DB2 engine uses, which the user does not control. This connection is a separate connection that can access the system catalog and issue commits to release locks. Utility procedures such as DATASETS and CONTENTS can cause this connection to be created, although other actions necessitate it as well. There is one connection of this type per libref, but it is not created until it is needed. If you have critical steps that must use the main SAS task connection for performance reasons, refrain from using the DEFER=YES option in the LIBNAME statement. It is possible that the utility connection can be established from that task, causing the connection that you use for your opening to be from a slower subtask.

In summary, no single value works best for the CONNECTION= option in all possible situations. You might need to try different values and arrange your SAS programs in different ways to obtain the best performance possible.

---

## Passing SAS Functions to DB2 under z/OS

SAS/ACCESS Interface to DB2 under z/OS passes the following SAS functions to DB2 for processing if the DBMS driver or client that you are using supports this

function. Where the DB2 function name differs from the SAS function name, the DB2 name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                  |                     |
|------------------|---------------------|
| ABS              | MOD (see note)      |
| ARCOS (ACOS)     | MONTH               |
| ARSIN (ASIN)     | QTR (QUARTER)       |
| ATAN             | RIGHT (RTRIM)       |
| ATAN2            | SECOND              |
| AVG              | SIGN                |
| CEIL             | SIN                 |
| COS              | SINH                |
| COSH             | SQRT                |
| COUNT (COUNTBIG) | STD (STDDEV)        |
| DAY              | STRIP               |
| EXP              | SUBSTR              |
| FLOOR            | SUM                 |
| HOUR             | TAN                 |
| INDEX (LOCATE)   | TANH                |
| LEFT (LTRIM)     | TRANWRD (REPLACE)   |
| LOWCASE (LCASE)  | TRIMN (RTRIM)       |
| LOG              | TRUNC               |
| LOG10            | UPCASE (UCASE)      |
| MAX              | VAR (VARIANCE)      |
| MIN              | WEEKDAY (DAYOFWEEK) |
| MINUTE           | YEAR                |

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to DB2. Due to incompatibility in date and time functions between DB2 and SAS, DB2 might not process them correctly. Check your results to determine whether these functions are working as expected.

|                 |                      |
|-----------------|----------------------|
| DATEPART (DATE) | TIMEPART (TIME)      |
| LENGTH          | TODAY (CURRENT DATE) |
| REPEAT          | TRANSLATE            |

The following functions are passed to the DBMS. They are not passed to the DBMS when you connect using [DRDA](#).

|       |
|-------|
| YEAR  |
| MONTH |
| DAY   |

---

# Passing Joins to DB2 under z/OS

With these exceptions, multiple libref joins are passed to DB2 z/OS.

- If you specify the SERVER= option for one libref, you must also specify it for the others, and its value must be the same for all librefs.
- If you specify the DIRECT\_SQL= option for one or multiple librefs, you must not set it to NO, NONE, or NOGENSQL.

For completeness, the portable code checks these options, regardless of the engine:

- DBCONINIT=
- DBCONTERM=
- DBLIBINIT=
- DBLIBTERM=
- DIRECT\_EXE=
- DIRECT\_SQL=
- PRESERVE\_COL\_NAMES=
- PRESERVE\_TAB\_NAMES=

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

# SAS System Options, Settings, and Macros for DB2 under z/OS

---

## System Options

You can use these SAS system options when you start a SAS session that accesses DB2 under z/OS.

**DB2DECPT=decimal-value**

specifies the value of the DB2 DECPOINT= option. The *decpoint-value* argument can be a period (.) or a comma (,). The default is a period (.).

DB2DECPT= is valid as part of the configuration file when you start SAS.

**DB2IN= 'database-name.tablespace-name' | 'DATABASE database-name'**

lets you specify the database and tablespace in which you want to create a new table. The DB2IN= option is relevant only when you are creating a new table. If

you omit this option, the default is to create the table in the default database and tablespace.

*database.tablespace* specifies the names of the database and tablespace.

'DATABASE *database-name*' specifies only the database name. Enclose the entire specification in single quotation marks.

You can override the DB2IN= system option with the IN= LIBNAME or data set option.

#### DB2PLAN=*plan-name*

specifies the name of the plan that is used when connecting (or binding) SAS to DB2. SAS provides the DBRM members that can be used to create a DB2 plan. The plan can be adapted for each user's site. The value for DB2PLAN= can be changed at any time during a SAS session, so that different plans can be used for different SAS steps. However, if you use more than one plan during a single SAS session, you must understand how and when SAS/ACCESS Interface to DB2 under z/OS makes the connections. If one plan is in effect and you specify a new plan, the new plan does not affect the existing DB2 connections.

#### DB2RRS | NODB2RRS

specifies the attachment facility to be used for a SAS session when connecting to DB2. This option is an invocation-only option.

Specify NODB2RRS, the default, to use the Call Attachment Facility (CAF).

Specify DB2RRS to use the Recoverable Resource Manager Services Attachment Facility (RRSAF). For details about using RRSAF, see ["How the Interface to DB2 Works" on page 841..](#)

#### DB2RRSMP | NODB2RRSMP

specifies that the multiphase SRRCMIT commit and SRRBACK rollback calls are used instead of the COMMIT and ROLLBACK SQL statements. This option is ignored unless DB2RRS is specified. This option is available only at invocation.

Specify NODB2RRSMP, the default, when DB2 is the only Resource Manager for your application. Specify DB2RRSMP when your application has other resource managers, which requires the use of the multiphase calls. Using the multiphase calls when DB2 is your only resource manager can have performance implications. Using COMMIT and ROLLBACK when you have more than one resource manager can result in an error, depending on the release of DB2.

#### DB2SSID=*subsystem-name*

specifies the DB2 subsystem name. The *subsystem-name* argument is one to four characters that consist of letters, numbers, or national characters (#, \$, or @); the first character must be a letter. The default value is DB2. For more information, see ["Values" on page 821.](#)

DB2SSID= is valid in the OPTIONS statement, as part of the configuration file, and when you start SAS.

You can override the DB2SSID= system option with the [SSID= connection option](#).

#### DB2UPD=Y | N

specifies whether the user has privileges through SAS/ACCESS Interface to DB2 under z/OS to update DB2 tables. This option applies only to the user's Update privileges through the interface and not necessarily to the user's privileges while using DB2 directly. Altering the value of DB2UPD= has no effect on your DBMS privileges, which have been specified with the GRANT statement. The default is Y (Yes).

DB2UPD= is valid in the OPTIONS statement, as part of the configuration file, and when you start SAS. This option does not affect PROC DBLOAD or the SAS 5 compatibility procedures.

## Values

To connect to DB2, you must specify a valid DB2 subsystem name in one of these ways.

- the DB2SSID= system option (SAS/ACCESS Interface to DB2 under z/OS uses this value if no DB2 subsystem is specified.)
- the SSID= option in the PROC ACCESS statement
- the SSID= statement of PROC DBLOAD
- the SSID= option in the PROC SQL CONNECT statement, which is part of the SQL pass-through facility
- the SSID= connection option in the LIBNAME statement

If a site does not specify a valid DB2 subsystem when it accesses DB2, this message is generated:

```
ERROR: Cannot connect to DB2 subsystem XXXX,
rc=12, reason code = 00F30006. See the
Call Attachment Facility documentation
for an explanation.
```

XXXX is the name of the subsystem to which SAS tried to connect. To find the correct value for your DB2 subsystem ID, contact your database administrator.

## Macros

Use the automatic SYSDBRC macro variable to capture return codes when using the DB2 engine. The macro variable is set to the last return code that was encountered when code execution took place using a SAS/ACCESS interface. If SAS/ACCESS Interface to DB2 under z/OS was the last usage of a SAS/ACCESS interface, then SYSDBRC contains the last DB2 return code. If you reference SYSDBRC before engine processing takes place, you receive this message:

```
WARNING: Apparent symbolic reference SYSDBRC not resolved.
```

Use SYSDBRC for conditional post-processing. Below is an example of how to abend a job. The table DB2TEST is dropped from DB2 after the view descriptor is created, resulting in a -204 code.

```
data test;
x=1;
y=2;
proc dbload dbms=db2 data=test;
table=db2test;
in 'database test';
load;
run;
```

```

proc access dbms=db2;
create work.temp.access;
table=user1.db2test;
create work.temp.view;
select all;
run;
proc sql;
execute(drop table db2test)by db2;
quit;

proc print data=temp;
run;

data _null_;
if "&sysdbrc" not in ('0','100') then
do;
put 'The DB2 Return Code is: ' "&sysdbrc";
abort abend;
end;
run;

```

Because an abnormal end to processing prevents the log from being captured, you must capture the SAS log by using the SAS system option, ALTLOG=. For more information, see [SAS System Options: Reference](#)

---

## Bulk Loading for DB2 under z/OS

---

### Overview

By default, the DB2 under z/OS interface loads data into tables by preparing an SQL INSERT statement, executing the INSERT statement for each row, and issuing a COMMIT statement. You must specify **BULKLOAD=YES** to start the DB2 LOAD utility. You can then bulk load rows of data as a single unit, which can significantly enhance performance. For smaller tables, the extra overhead of the bulk-loading process might slow performance. For larger tables, the speed of the bulk-loading process outweighs the overhead costs.

When you use bulk loading, see the SYSPRINT output for information about the load. If you run the LOAD utility and it fails, ignore the messages in the SAS log because they might be inaccurate. However, if errors existed before you ran the LOAD utility, error messages in the SAS log might be valid.

SAS/ACCESS Interface to DB2 under z/OS provides bulk loading through DSNUTILS, an IBM stored procedure that starts the DB2 LOAD utility. Because the LOAD utility is complex, familiarize yourself with it before you use it through SAS/ACCESS. Also check with your database administrator to determine whether this utility is available.

---

## Data Set Options for Bulk Loading

The DB2 under z/OS engine supports the following bulk-load data set options. All begin with BL\_ for bulk loading. To use the bulk-load facility, you must specify **BULKLOAD=YES** or all bulk-load options are ignored. (The DB2 under z/OS interface alias for BULKLOAD= is DB2LDUTIL=.)

- [BL\\_DB2CURSOR=](#)
- [BL\\_DB2DATACLAS=](#)
- [BL\\_DB2DEVT\\_PERM=](#)
- [BL\\_DB2DEVT\\_TEMP=](#)
- [BL\\_DB2DISC=](#)
- [BL\\_DB2ERR=](#)
- [BL\\_DB2IN=](#)
- [BL\\_DB2LDCT1=](#)
- [BL\\_DB2LDCT2=](#)
- [BL\\_DB2LDCT3=](#)
- [BL\\_DB2LDEXT=](#)
- [BL\\_DB2MAP=](#)
- [BL\\_DB2MGMTCLAS=](#)
- [BL\\_DB2PRINT=](#)
- [BL\\_DB2PRNLOG=](#)
- [BL\\_DB2REC=](#)
- [BL\\_DB2RECSP=](#)
- [BL\\_DB2RSTRT=](#)
- [BL\\_DB2SPC\\_PERM=](#)
- [BL\\_DB2SPC\\_TEMP=](#)
- [BL\\_DB2STORCLAS=](#)
- [BL\\_DB2TBLXST=](#)
- [BL\\_DB2UNITCOUNT=](#)
- [BL\\_DB2UTID=](#)
- [BULKLOAD=](#)

---

## File Allocation and Naming for Bulk Loading

When you use bulk loading, these files (data sets) are allocated.

- The DB2 DSNUTILS procedure allocates these as new and catalogs the SysDisc, SysMap, and SysErr files unless `BL_DB2LDEXT=USERUN`. If `BL_DB2LDEXT=USERUN`, data sets are allocated as old and are kept.
- The DB2 interface engine allocates as new and catalogs the files SysIn and SysRec when the execution method specifies to generate them.
- The DB2 interface engine allocates as new and catalogs the file SysPrint when the execution method specifies to run the utility.

All allocations of these data sets are reversed by the end of the step. If errors occur before SysRec is generated, any of these data sets that were allocated as new and cataloged are deleted as part of cleanup because they would be empty.

The interface engine uses these options when it allocates nonexisting SYS data set names.

- DSNUTILS uses `BL_DB2DEVT_PERM=` and `BL_DB2SPC_PERM=` for SysDisc, SysMap, and SysErr.
- The DB2 interface engine uses `BL_DB2DEVT_PERM=` for SysIn, SysRec, and SysPrint.
- SysRec uses `BL_DB2RECSPC=`. `BL_DB2RECSPC=` is necessary because the engine cannot determine how much space the SysRec requires—it depends on the volume of data being loaded into the table.
- DSNUTILS uses `BL_DB2DEVT_TEMP=` and `BL_DB2SPC_TEMP=` to allocate the other data set names that the LOAD utility requires.

This table shows how SysIn and SysRec are allocated based on the values of `BL_DB2LDEXT=` and `BL_DB2IN=`, and `BL_DB2REC=`.

*Table 19.3 SysIn and SysRec Allocation*

| <code>BL_DB2LDEXT=</code> | <code>BL_DB2IN=</code> /<br><code>BL_DB2REC=</code> | Data set<br>name | DISPOSITION          |
|---------------------------|-----------------------------------------------------|------------------|----------------------|
| GENRUN                    | not specified                                       | generated        | NEW, CATALOG, DELETE |
| GENRUN                    | specified                                           | specified        | NEW, CATALOG, DELETE |
| GENONLY                   | not specified                                       | generated        | NEW, CATALOG, DELETE |
| GENONLY                   | specified                                           | specified        | NEW, CATALOG, DELETE |
| USERUN                    | not specified                                       | ERROR            |                      |
| USERUN                    | specified                                           | specified        | OLD, KEEP, KEEP      |

When SAS/ACCESS Interface to DB2 under z/OS uses existing files, you must specify the file names. When the interface generates the files, it creates them with names that you provide or with unique names that it generates. Engine-generated file names use system-generated data set names with the format `SYSyyddd.Thhmmss.RA000.jobname.name.Hgg`, where

`SYSyyddd`

is replaced by the user ID. The user ID that is used to prequalify these generated data set names is determined the same way as within the rest of SAS, except

when running in a server environment. In a server environment, the authenticated ID of the client is used.

*name*

is replaced by the given SYS ddname of the data set.

For example, if you do not specify any data set names and run GENRUN under TSO, you obtain a set of files allocated with names such as these:

```
USERID.T125547.RA000.USERID.DB2DISC.H01
USERID.T125547.RA000.USERID.DB2ERR.H01
USERID.T125547.RA000.USERID.DB2IN.H01
USERID.T125547.RA000.USERID.DB2MAP.H01
USERID.T125547.RA000.USERID.DB2PRINT.H01
USERID.T125547.RA000.USERID.DB2REC.H01
```

This naming convention produces unique names, even within a sysplex (within one second per user ID per system). It therefore makes it easy to associate all information for each utility execution and separate it from other executions.

Bulk-load files are removed at the end of the load process to save space. They are not removed if the utility fails to allow for the load process to be restarted.

## Bulk Loading Examples

Use these LIBNAME statements for all examples.

```
libname db2lib db2;
libname shlib db2 connection=shared;
```

Create a table.

```
data db2lib.table1 (bulkload=yes);
x=1;
name='Tom';
run;
```

Append Table1 to itself.

```
data shlib.table1
(bulkload=yes bl_db2tblxst=yes bl_db2ldct1='RESUME YES');
set shlib.table1;
run;
```

Replace Table1 with itself.

```
data shlib.table1
(bulkload=yes bl_db2tblxst=yes bl_db2ldct1='REPLACE');
set shlib.table1;
run;
```

Load DB2 tables directly from other objects.

```
data db2lib.emp (bulkload=yes
bl_db2ldct1='REPLACE LOG NO NOCOPYPEND'
bl_db2cursor='select * from dsn8710.emp');
set db2lib.emp (obs=0);
run;
```

You can also use this option in a PROC SQL statement to load DB2 tables directly from other objects, as shown below.

```

options sastrace=',,d';
libname db2lib db2 authid=dsn8710;
libname mylib db2;

proc datasets library=mylib;
  delete emp;run;

proc sql;
  connect to db2;
  create table mylib.emp
    (BULKLOAD=YES
     BL_DB2LDCT1='REPLACE LOG NO NOCOPYPEND'
     BL_DB2CURSOR='SELECT FIRSTNAME, LASTNAME, WORKDEPT,
                   HIREDATE, JOB, SALARY, BONUS, COMM
                   FROM DSN8710.EMP')
  as select firstname, lastname, workdept,
           hiredate, job, salary, bonus, comm
  from db2lib.emp (obs=0);
quit;

```

Here is another similar example.

```

options sastrace=',,d';
libname db2lib db2 authid=dsn8710;
libname mylib db2;

proc datasets library=mylib;
  delete emp;run;

proc sql;
  connect to db2;
  create table mylib.emp
    (BULKLOAD=YES
     BL_DB2LDCT1='REPLACE LOG NO NOCOPYPEND'
     BL_DB2CURSOR='SELECT FIRSTNAME, LASTNAME, WORKDEPT,
                   HIREDATE, JOB, SALARY, BONUS, COMM
                   FROM DSN8710.EMP'
     BL_DB2LDCT3='RUNSTATS TABLESPACE DSNDB04.TEMPTTABL
                  TABLE(ALL) INDEX(ALL) REPORT YES')
  as select firstname, lastname, workdept,
           hiredate, job, salary, bonus, comm
  from db2lib.emp (obs=0);
quit;

```

Generate control and data files, create the table, but do not run the utility to load it.

```

data shlib.table2 (bulkload=yes
  bl_db2ldext=genonly bl_db2in='userid.sysin'
  bl_db2rec='userid.sysrec');
  set shlib.table1;
run;

```

Use the control and data files that you generated in the preceding example load the table. The OBS=1 data set option on the input file prevents the DATA step from reading the whole file. Because the data is really in SysRec, you need only the input file to satisfy the engine.

```
data db2lib.table2 (bulkload=yes bl_db2tblxst=yes
```

```

      bl_db2ldext=userun bl_db2in='userid.sysin'
      bl_db2rec='userid.sysrec');
      set db2lib.table1 (obs=1);
run;

```

A more efficient approach than the previous example is to eliminate going to DB2 to read even one observation from the input table. This also means that the DATA step processes only one observation, without any input I/O. Note that the one variable V is not on the table. Any variables listed here (there is no need for more than one), are irrelevant because the table already exists; they are not used.

```

data db2lib.table2 (bulkload=yes bl_db2tblxst=yes
      bl_db2ldext=userun bl_db2in='userid.sysin'
      bl_db2rec='userid.sysrec');
      v=0;
run;

```

Generate control and data files, but do not create the table or run the utility. Specifying BL\_DB2TBLXST=YES when the table does not exist prevents you from creating the table; this only makes sense because you are not going to load any data into the table at this time.

```

data db2lib.table3 (bulkload=yes bl_db2tblxst=yes
      bl_db2ldext=genonly bl_db2in='userid.sysin'
      bl_db2rec='userid.sysrec');
      set db2lib.table1;
run;

```

Use the control and data files that you generated in the preceding example to load the table. The OBS=1 data set option on the input file prevents the DATA step from reading the whole file. In this case, you must specify the input file because it contains the column definitions that are necessary to create the table.

```

data shlib.table3 (bulkload=yes bl_db2ldext=userun
      bl_db2in='userid.sysin' bl_db2rec='userid.sysrec');
      set shlib.table1 (obs=1);
run;

```

If you know the column names, a more efficient approach than the previous example is to eliminate going to DB2 to obtain the column definitions. In this case, the variable names and data types must match, because they are used to create the table. However, the values specified for the variables are not included on the table, because all data to load comes from the existing SysRec.

```

data db2lib.table3 (bulkload=yes bl_db2ldext=userun
      bl_db2in='userid.sysin' bl_db2rec='userid.sysrec');
      x=0;
      name='????';
run;

```

You can use other applications that process output.

```

data work.a;
      x=1;
run;

proc sql;
      create db2lib.table4 (bulkload=yes) as select * from a;
quit;

```

# Locking in the DB2 under z/OS Interface

The following LIBNAME and data set options let you control how the DB2 under z/OS interface handles locking. For general information about an option, see ["LIBNAME Statement: External Databases" on page 129](#). For additional information, see your DB2 documentation.

```
READ_LOCK_TYPE=TABLE
UPDATE_LOCK_TYPE=TABLE
READ_ISOLATION_LEVEL= CS | UR | RR | "RR KEEP UPDATE LOCKS" | RS |
"RS KEEP UPDATE LOCKS"
```

Here are the valid values for this option. DB2 determines the default isolation level.

*Table 19.4 Isolation Levels for DB2 under z/OS*

| Value                             | Isolation Level                   |
|-----------------------------------|-----------------------------------|
| CS                                | Cursor stability                  |
| UR                                | Uncommitted read                  |
| RR                                | Repeatable read                   |
| RR KEEP UPDATE LOCKS <sup>1</sup> | Repeatable read keep update locks |
| RS                                | Read stability                    |
| RS KEEP UPDATE LOCKS <sup>1</sup> | Read stability keep update locks  |

<sup>1</sup> When specifying a value that consists of multiple words, enclose the entire string in quotation marks.

UPDATE\_ISOLATION\_LEVEL= CS | UR | RR | "RR KEEP UPDATE LOCKS" | RS | "RS KEEP UPDATE LOCKS"

The valid values for this option are described in the preceding table. The default isolation level is determined by DB2.

---

# Naming Conventions for DB2 under z/OS

---

## Overview

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

DB2 objects include tables, views, columns, and indexes. They follow these naming conventions.

- These objects must have names of the following length in characters: column (1–30), index (1–128), table (1–128), view (1–128), alias (1–128), synonym (1–128), correlation (1–128). However, SAS limits table names to 32 bytes. This limitation prevents database table objects that are specified through a DATA step—for example, to have names that are longer than 32.

These objects must have names from 1–8 characters long: authorization ID, referential constraint, database, tablespace, storage group, package, or plan.

A location name can be 1–16 characters long.

- A name must begin with a letter. If the name is in quotation marks, it can start with and contain any character. Depending on how your string delimiter is specified, quoted strings can contain quotation marks (for example, “O’Malley”).
- A name can contain the letters A–Z, numbers from 0–9, number or pound sign (#), dollar sign (\$), or at symbol (@).
- Names are not case sensitive. For example, CUSTOMER and Customer are the same. However, if the name of the object is in double quotation marks, it is case sensitive.
- A name cannot be a reserved word in DB2.
- A name cannot be the same as another DB2 object. For example, each column name within the same table must be unique.

---

## Case Sensitivity and Special Characters in Object Names

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Although DB2 is case sensitive, it converts table and column names to uppercase by default. To preserve the case of the table and column names that you send to DB2, enclose them in double quotation marks.

## Using Table Aliases or Synonyms

DB2 under z/OS supports the use of table aliases or table synonyms. If you use aliases or synonyms in your database, then you must specify the qualifying value to use when accessing the alias or synonym name, even if the qualifying value is NULL. The **AUTHID=** and **SCHEMA= LIBNAME** options specify qualifying values for tables.

To specify a NULL value for the AUTHID= LIBNAME option, submit this code:

```
libname mylib2 db2 authid=''; /* no space between quotation marks */
```

To specify a NULL value for the SCHEMA= LIBNAME option, submit this code:

```
libname mylib db2 schema=''; /* no space between quotation marks */
```

You can also specify NULL values for the **AUTHID=** and **SCHEMA=** data set options.

---

## Data Types for DB2 under z/OS

### Overview

Every column in a table has a name and a data type. The data type tells DB2 how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about DB2 data types, NULL and default values, and data conversions.

For more information about DB2 data types, see your DB2 SQL reference documentation.

SAS/ACCESS does not support some types of distinct DB2 data types.

---

## Supported Data Types for DB2 under z/OS

Here are the data types that DB2 under z/OS supports:

- Character data:

|                                             |                        |
|---------------------------------------------|------------------------|
| CHAR( <i>n</i> )                            | LONG VARCHAR           |
| CLOB (character large object)               | LONG VARGRAPHIC        |
| DBCLOB (double-byte character large object) | VARCHAR( <i>n</i> )    |
| GRAPHIC( <i>n</i> )                         | VARGRAPHIC( <i>n</i> ) |

- Numeric data:

|                                                          |                                            |
|----------------------------------------------------------|--------------------------------------------|
| DECIMAL( <i>p,s</i> )   DEC( <i>p,s</i> )                | INTEGER   INT, REAL  <br>FLOAT( <i>n</i> ) |
| FLOAT( <i>n</i> )   DOUBLE PRECISION   FLOAT<br>  DOUBLE | SMALLINT                                   |

Even though the DB2 numeric columns have these distinct data types, the DB2 engine accesses, inserts, and loads all numerics as FLOATs.

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- 
- Date, time, and timestamp data:

DATE    TIMESTAMP  
TIME

DB2 date and time data types are similar to SAS date and time values in that they are stored internally as numeric values and are displayed in a site-chosen format. The DB2 data types for dates, times, and timestamps are listed here. Note that columns of these data types might contain data values that are out of range for SAS, which handles dates from 1582 A.D. through 20,000 A.D.

- Binary data: BLOB (binary large object)

---

**Note:** Support for large object data was added in [SAS 9.4M2](#).

- 
- ROWID data type

For more information about DB2 data types, see your DB2 documentation.

---

## DB2 Null and Default Values

DB2 has a special value that is called NULL. A DB2 NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a DB2 NULL value, it interprets it as a SAS missing value.

DB2 columns can be specified so that they do not allow NULL data. For example, NOT NULL would indicate that DB2 does not allow a row to be added to the TestID.Customers table unless there is a value for CUSTOMER. When creating a DB2 table with SAS/ACCESS, you can use the [DBNULL=](#) data set option to indicate whether NULL is a valid value for specified columns.

You can also define DB2 columns as NOT NULL WITH DEFAULT. The following table lists default values that DB2 assigns to columns that you define as NOT NULL WITH DEFAULT. An example of such a column is STATE in Testid.Customers. If a column is omitted from a view descriptor, default values are assigned to the column.

However, if a column is specified in a view descriptor and it has no values, no default values are assigned.

**Table 19.5** Default Values That DB2 Assigns for Columns Specified as NOT NULL WITH DEFAULT

| DB2 Column Type                                       | DB2 Default <sup>1</sup>                            |
|-------------------------------------------------------|-----------------------------------------------------|
| CHAR( <i>n</i> )   GRAPHIC( <i>n</i> )                | blanks, unless the NULLCHARVAL= option is specified |
| VARCHAR   LONG VARCHAR   VARGRAPHIC   LONG VARGRAPHIC | empty string                                        |
| BLOB   CLOB   DBCLOB                                  | empty string                                        |
| SMALLINT   INT   FLOAT   DECIMAL   REAL               | 0                                                   |
| DATE                                                  | current date, derived from the system clock         |
| TIME                                                  | current time, derived from the system clock         |
| TIMESTAMP                                             | current timestamp, derived from the system clock    |

<sup>1</sup> The default values that are listed in this table pertain to values that DB2 assigns.

Knowing whether a DB2 column allows NULL values or whether DB2 supplies a default value can assist you in writing selection criteria and in entering values to update a table. Unless a column is specified as NOT NULL or NOT NULL WITH DEFAULT, the column allows NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how the DBMS handles SAS missing character values, use the **NULLCHEAR=** and **NULLCHARVAL=** data set options.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to DB2 assigns to SAS variables when using the **LIBNAME** statement to read from a DB2 table. These default formats are based on DB2 column attributes.

**Table 19.6** LIBNAME Statement: Default SAS Formats for DB2 Data Types

| DB2 Column Type               | Default SAS Format |
|-------------------------------|--------------------|
| CHAR( <i>n</i> ) <sup>1</sup> | \$ <i>w</i> .      |

| DB2 Column Type                | Default SAS Format    |
|--------------------------------|-----------------------|
| VARCHAR( $n$ ) <sup>1</sup>    |                       |
| LONG VARCHAR                   |                       |
| BLOB                           | \$HEX32767.           |
| CLOB                           | \$32767.              |
| DBCLOB                         | \$w.                  |
| GRAPHIC( $n$ ) <sup>1</sup>    | \$w. ( $w \leq 127$ ) |
| VARGRAPHIC( $n$ ) <sup>1</sup> | \$127. ( $w > 127$ )  |
| LONG VARGRAPHIC                |                       |
| INTEGER                        | 11.                   |
| SMALLINT                       | 6.                    |
| DECIMAL( $m,n$ ) <sup>2</sup>  | w+2.d                 |
| FLOAT                          | none                  |
| DOUBLE PRECISION               | none                  |
| REAL                           | none                  |
| NUMERIC( $m,n$ ) <sup>2</sup>  | w+2.d                 |
| DATE                           | DATE9.                |
| TIME                           | TIME8.                |
| TIMESTAMP                      | DATETIME30.6          |
| ROWID                          | none                  |

1  $n$  in DB2 character and graphic data types is equivalent to  $w$  in SAS formats.

2  $m$  and  $n$  in DB2 numeric data types are equivalent to  $w$  and  $d$  in SAS formats.

This table shows the default DB2 data types that SAS/ACCESS assigns to SAS variable formats during output operations.

**Table 19.7 LIBNAME Statement: Default DB2 Data Types for SAS Variable Formats**

| SAS Variable Format | DB2 Data Type                           |
|---------------------|-----------------------------------------|
| \$w.                | CHARACTER( $n$ ) for 1–255 <sup>1</sup> |
| \$CHARw.            | VARCHAR( $n$ ) for >255 <sup>1</sup>    |
| \$VARYINGw.         |                                         |
| \$HEXw.             |                                         |

| SAS Variable Format       | DB2 Data Type |
|---------------------------|---------------|
| any date format           | DATE          |
| any time format           | TIME          |
| any datetime format       | TIMESTAMP     |
| all other numeric formats | FLOAT         |

<sup>1</sup> *n* in DB2 character and graphic data types is equivalent to *w* in SAS formats.

## ACCESS Procedure Data Conversions

This table shows the default SAS variable formats that SAS/ACCESS assigns to DB2 data types when you use the [ACCESS procedure](#).

*Table 19.8 ACCESS Procedure: Default SAS Formats for DB2 Data Types*

| DB2 Column Type        | Default SAS Format                                           |
|------------------------|--------------------------------------------------------------|
| CHAR( <i>n</i> )       | \$ <i>w</i> . ( <i>w</i> <=199) <sup>1</sup>                 |
| VARCHAR( <i>n</i> )    | \$ <i>w</i> .<br>\$200. ( <i>w</i> >200) <sup>1</sup>        |
| LONG VARCHAR           | \$ <i>w</i> .                                                |
| BLOB                   | none                                                         |
| CLOB                   | none                                                         |
| DBCLOB                 | none                                                         |
| GRAPHIC( <i>n</i> )    | \$ <i>w</i> . ( <i>w</i> <=127) <sup>1</sup>                 |
| VARGRAPHIC( <i>n</i> ) | \$127. ( <i>w</i> >127) <sup>1</sup>                         |
| LONG VARGRAPHIC        |                                                              |
| INTEGER                | 11.                                                          |
| SMALLINT               | 6.                                                           |
| DECIMAL( <i>m,n</i> )  | <i>w+2.d</i><br>For example, DEC(6,4) becomes SAS format 8.4 |
| REAL                   | E12.                                                         |

| DB2 Column Type       | Default SAS Format                                                             |
|-----------------------|--------------------------------------------------------------------------------|
| DOUBLE PRECISION      | E12.                                                                           |
| FLOAT( <i>n</i> )     | E12.                                                                           |
| FLOAT                 | E12.                                                                           |
| NUMERIC( <i>m,n</i> ) | <i>w+d</i> <sup>2</sup><br>For example, NUMERIC(6,2) becomes<br>SAS format 8.2 |
| DATE                  | DATE7.                                                                         |
| TIME                  | TIME8.                                                                         |
| TIMESTAMP             | DATETIME30.6                                                                   |

<sup>1</sup> *n* in DB2 character and graphic data types is equivalent to *w* in SAS formats.

<sup>2</sup> *m* and *n* in DB2 numeric data types are equivalent to *w* and *d* in SAS formats.

You can use the YEARCUTOFF= option to make your DATE7. dates comply with Year 2000 standards. For more information about this SAS system option, see [SAS System Options: Reference](#).

## DBLOAD Procedure Data Conversions

This table shows the default DB2 data types that SAS/ACCESS assigns to SAS variable formats when you use the [DBLOAD procedure](#).

**Table 19.9 DBLOAD Procedure: Default DB2 Data Types for SAS Variable Formats**

| SAS Variable Format  | DB2 Data Type                      |
|----------------------|------------------------------------|
| \$ <i>w</i> .        | CHARACTER( <i>n</i> )              |
| \$CHAR <i>w</i> .    |                                    |
| \$VARYING <i>w</i> . |                                    |
| \$HEX <i>w</i> .     |                                    |
| any date format      | DATE                               |
| any time format      | TIME                               |
| any datetime format  | TIMESTAMP                          |
| <i>w.d</i>           | DECIMAL( <i>m,n</i> ) <sup>1</sup> |
| IB, IBR, PIB, PIBR   | INTEGER                            |
|                      | FLOAT                              |

| SAS Variable Format                                                                          | DB2 Data Type |
|----------------------------------------------------------------------------------------------|---------------|
| all other numeric formats                                                                    |               |
| <small>1 m and n in DB2 numeric data types are equivalent to w and d in SAS formats.</small> |               |

## Temporal Data for DB2 under z/OS

### Overview of Temporal Data for DB2 under z/OS

Temporal data stores information about values that change over time. Temporal data is structured to reduce the effort that is needed to maintain and manage temporal columns in database tables. Using temporal data enables you to take advantage of simplified query syntax and semantics when working with records that contain data that pertains to specific time periods.

In DB2, there are three types of temporal data: system time, business time, and bitemporal data. *System time* tracks when changes are made to the state of data in a table. The state of a record in a table is determined by the SYS\_START and SYS\_END values. In a table that manages system time, the transaction time, or TRANS\_START, is also maintained. The TRANS\_START time records when a transaction to change a record, such as updating the SYS\_END date, took place. SAS generates these three columns automatically when you create a table that stores system time data.

*Business time* tracks the effective dates for business data, such as promotional start and end dates. The effective start and end dates for business time data are stored in the BUS\_START and BUS\_END values. SAS generates these columns automatically when you create a table that stores business time data.

*Bitemporal data* tracks system time data and business time data. A table that contains bitemporal data contains all of the columns that are automatically included for system time data and business time data: SYS\_START, SYS\_END, TRANS\_START, BUS\_START, and BUS\_END.

### System Time and Bitemporal Data: History Tables

Tables that contain business time data include historical, current, and future data records. For tables that contain system time data or bitemporal data, SAS automatically generates history tables. The name of a history table is constructed as `<base-table>_HISTORY`. History tables contain records that are determined to be historical. That is, when the SYS\_END date precedes the current date or datetime, then the record is a historical record. SAS/ACCESS for DB2 under z/OS automatically issues an `ALTER TABLE ... ADD VERSIONING` statement that links the base table with its history table.

The temporal table and its associated history table must be stored in separate tablespaces. In addition, each table should be the only table in its tablespace. If a tablespace is not included as part of the creation of a temporal table from a SAS DATA step, then DB2 automatically assigns a unique tablespace to the temporal table and to the associated history table. For this reason, you cannot specify a tablespace in the values for the IN= option or the DB2IN= system option, if you are working with temporal tables that contain system time or bitemporal data. For more information, see [IN= data set option](#), [IN= LIBNAME option](#), or “[SAS System Options, Settings, and Macros for DB2 under z/OS](#)” on page 819.

## Data Set Options for Temporal Data

The following data set options are used when you work with temporal data:

### TEMPORAL=

specifies the type of temporal data that is stored in a table. Possible values are BUSINESS, SYSTEM, or BITEMPORAL. For more information, see “[TEMPORAL= Data Set Option](#)” on page 614.

### BUSINESS\_DATATYPE=

specifies the data type for the BUS\_START and BUS\_END values. Possible values are DATE or TIMESTAMP(6). The default value is TIMESTAMP(6). For more information, see “[BUSINESS\\_DATATYPE= Data Set Option](#)” on page 494.

### BUSINESS\_TIMEFRAME=

specifies a time period to be used when querying or modifying a table that contains temporal data. You provide the beginning and ending values for the time period based on the BUSINESS\_DATATYPE value. Specify the time period in the format:

```
FROM <date-or-datetime> TO <date-or-datetime>
```

For example, use the following code to specify the time period from January 1, 2014 to December 31, 2014 (using date values):

```
busines_timeframe="from '01JAN2014'd to '31DEC2014'd"
```

For more information, see “[BUSINESS\\_TIMEFRAME= Data Set Option](#)” on page 495.

### SYSTEM\_TIMEFRAME=

specifies a time period to be used when querying or modifying a table that contains temporal data. You provide the beginning and ending datetime values in the format:

```
FROM <datetime> TO <datetime>
```

Provide datetime values that include year, month, day, hours, minutes, seconds, and fractions of a second. For example, use the following code to specify the time period from midnight, November 1, 2012 to midnight, December 1, 2030:

```
system_timeframe="from '2012-11-01-00.00.00.0' to '2030-12-01-00.00.00.0'"
```

For more information, see “[SYSTEM\\_TIMEFRAME= Data Set Option](#)” on page 612.

### OVERLAPS=

specifies columns that should not contain active business time periods that overlap. By default, overlaps are allowed for active business time records.

Separate column names with commas. For more information, see “[OVERLAPS= Data Set Option](#)” on page 578.

# Understanding DB2 under z/OS Client/Server Authorization

## Libref Connections

When you use the DB2 interface, you can enable each client to control its own connections using its own authority—instead of sharing connections with other clients—by using the DB2 Recoverable Resource Manager Services Attachment Facility (RRSAF). See “[DB2 Attachment Facilities \(CAF and RRSAF\)](#)” on page 843 for information about this facility.

When you use SAS/ACCESS Interface to DB2 under z/OS with RRSAF, the authorization mechanism works differently than it does in Base SAS:

- In Base SAS, the SAS server always validates the client's authority before allowing the client to access a resource.
- In SAS/ACCESS Interface to DB2 under z/OS (with RRSAF), DB2 checks the authorization identifier that is carried by the connection from the SAS server. In most situations, this is the client's authorization identifier. In one situation, however, this is the SAS server's authorization identifier. A client can access a resource by using the server's authorization identifier only if the client uses a libref that was specified in the server session.

In this next example, a user assigns the libref SRVPRELIB in the SRV1 server session. In the client session, a user then issues a LIBNAME statement that makes a logical assignment using the libref MYPRELIB, and the user specifies the LIBNAME option SERVER=srv1. The client can then access resources by using the server's authority for the connection.

**1 In the server session**

```
libname srvpreflib db2 ssid=db25;
proc server id=srv1;
run;
```

**2 In the client session**

```
libname mypreflib server=srv1 sllibref=srvpreflib;
proc print data=mypreflib.db2table;
run;
```

In this example, because the client specifies a regular libref, MYDBLIB, the client has its own authority for the connections.

**1 In the server session**

```
libname mypreflib db2 ssid=db25;
```

```
proc server id=srv1;
run;
```

## 2 In the client session

```
libname mydblib server=srv1 roptions='ssid=db25' rengine=db2;
proc print data=mydblib.db2table;
run;
```

In this table, SAS/SHARE clients use LIBNAME statements to access SAS libraries and DB2 data through the server. In this description, a *logical* LIBNAME statement is a statement that associates a libref with another libref that was previously assigned.

**Table 19.10 Librefs and Their Authorization Implications**

---

### Client Session

---

```
libname local v8
'SAS.library' disp=old;
libname dblocal db2
connection=unique;
```

These statements execute in the client session. These are local assignments. The authority ID is the ID of the client.

```
libname remote 'SAS.library'
server=srv1 rengine=v8
roptions='disp=old';
libname dbremote
server=srv1 rengine=db2
roptions='connection=unique'
; 
```

These statements execute in the server session on behalf of the client. Libref Remote is a Base SAS engine remote assignment. Libref DbRemote is a DB2 engine remote assignment. In both cases, the authority ID is the ID of the client.

---

### Server Session (id=serv1)

---

```
libname predef v8
'SAS.library' disp=old;
libname dbpredef db2
connection=unique;
```

Because librefs PreDef and DbPreDef are specified in the server session, they can be referenced only by a client using a logical LIBNAME statement. There is no authority ID because clients cannot access these librefs directly.

---

### Logical Assignments - Client Session

---

```
libname alias (local);
libname dbalias (dblocal);
```

These statements create aliases ALIAS and DBALIAS for librefs Local and DbLocal, which were assigned in the client session above. The authority ID is the ID of the client.

```
libname logic server=srv1
slibref=predef;
libname dblogic server=srv1
slibref=dbpredef;
```

These statements refer to librefs PreDef and DbPreDef, which were assigned in the server session above. Libref Logic is a Base SAS engine logical assignment of remote libref PreDef. The authority ID for libref Logic is the ID of the client.

Libref DbLogic is a DB2 engine logical assignment of remote libref DbPreDef. The authority ID for libref DbLogic is the ID of the server.

---

For the Base SAS engine Remote and Logic librefs, it is the client's authority that is verified. (This is true for all Base SAS engine assignments.) DbRemote and DbLogic DB2 engine librefs refer to the same resources. However, it is the client's authority that is verified for DbRemote, whereas it is the server's authority that is verified for DbLogic. When using the DB2 interface, you can determine whether to use the client's authority or the server's authority to access DB2 data.

## Non-Libref Connections

When you make connections using the SQL pass-through facility or view descriptors, the connections to the database are not based on a DB2 engine libref. A connection that is created in the server, by using these features from a client, always has the authority of the client, because there is no server-established connection to reference.

This example uses the SAS/SHARE Remote SQL pass-through facility. The client has its own authority for the connections.

**1 In the server session:**

```
proc server id=srv1;
run;
```

**2 In the client session**

```
proc sql;
connect to remote (server=srv1 dbms=db2 dbmsarg=(ssid=db25));
select * from connection to remote
  (select * from db2table);
disconnect from remote;
quit;
```

This example uses a previously created view descriptor. The client has its own authority for the connections. The PreLib libref PreLib that was previously assigned and the client-assigned libref MyLib have no relevant difference. These are Base SAS engine librefs and not DB2 engine librefs.

**1 In the server session**

```
libname prelib V8 'SAS.library';
proc server id=srv1;
run;
```

**2 In the client session**

```
libname prelib server=srv1;
proc print data=prelib.accview;
run;
```

**3 In the client session**

```
libname mylib 'SAS.library2' server=srv1 rengine=v8;
```

```
proc print data=mylib.accview;
run;
```

---

## Known Issues with RRSAF Support

SAS/SERVE can use various communication access methods to communicate with clients. You can specify these through the COMAMID and COMAUX1 system options.

When you use XMS (Cross Memory Services) as an access method, DB2 also uses XMS in the same address space. Predefining DB2 server librefs before starting PROC SERVER can result in errors due to the loss of the XMS Authorization Index, because both SAS and DB2 are acquiring and releasing it. When using XMS as an access method, use only client-assigned librefs on the server.

This problem does not occur when you use the TCP/IP access method. So if you use TCP/IP instead of XMS, you can use both client-assigned (client authority) and server-preassigned (server authority) librefs. You can also use either access method if your connection is not based on a libref (client authority).

---

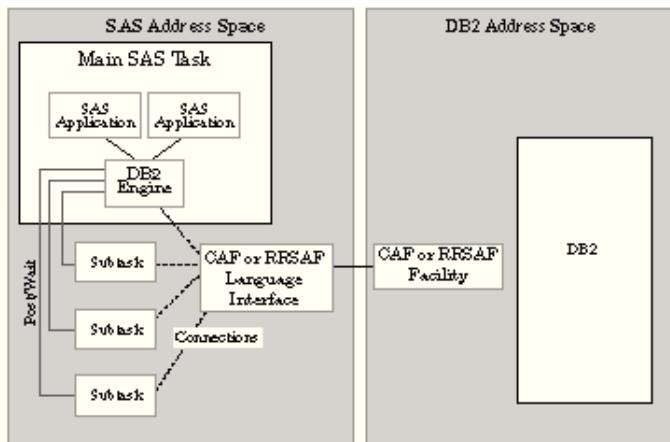
# DB2 under z/OS Information for the Database Administrator

---

## How the Interface to DB2 Works

SAS/ACCESS Interface to DB2 under z/OS uses either the Call Attachment Facility (CAF) or the Recoverable Resource Management Services Attachment Facility (RRSAF) to communicate with the local DB2 subsystem. Both attachment facilities enable programs to connect to DB2 and to use DB2 for SQL statements and commands. SAS/ACCESS Interface to DB2 under z/OS uses the attachment facilities to establish and control its connections to the local DB2 subsystem. DB2 allows only one connection for each task control block (TCB), or task. SAS and SAS executables run under one TCB, or task.

The DB2 LIBNAME statement lets SAS users connect to DB2 more than once. Because the CAF and RRSAF allow only one connection per TCB, SAS/ACCESS Interface to DB2 under z/OS attaches a subtask for each subsequent connection that is initiated. It uses the ATTACH, DETACH, POST, and WAIT assembler macros to create and communicate with the subtasks. It does not limit the number of connections or subtasks that a single SAS user can initiate. This image illustrates how the DB2 engine works.

**Figure 19.1** Design of the DB2 Engine

## How and When Connections Are Made

SAS/ACCESS Interface to DB2 under z/OS always makes an explicit connection to the local DB2 subsystem (SSID). When a connection executes successfully, a thread to DB2 is established. For each thread's or task's connection, DB2 establishes authorization identifiers (AUTHIDs).

The DB2 interface determines when to make a connection to DB2 based on the type of Open mode that a SAS application requests for the DB2 tables. The Open mode can be Read, Update, or Output. Here is the default behavior.

- SAS/ACCESS Interface to DB2 under z/OS shares the connection for all openings in Read mode for each DB2 LIBNAME statement
- SAS/ACCESS Interface to DB2 under z/OS acquires a separate connection to DB2 for every opening in Update or Output mode.

You can change this default behavior by using the **CONNECTION=** option.

Several SAS applications require SAS/ACCESS Interface to DB2 under z/OS to query the DB2 system catalogs. When this type of query is required, the DB2 interface acquires a separate connection to DB2 to avoid contention with other applications that are accessing the DB2 system catalogs. See “[Accessing DB2 System Catalogs](#)” on page 844 for more information.

The **DEFER= LIBNAME** option also controls when a connection is established. **UTILCONN\_TRANSIENT=** also allows control of the utility connection—namely, whether it must stay open.

## DDF Communication Database

DB2 Distributed Data Facility (DDF) Communication Database (CDB) enables DB2 z/OS applications to access data on other systems. Database administrators are responsible for customizing CDB. SAS/ACCESS Interface to DB2 under z/OS

supports both types of DDF: system-directed access (private protocol) and Distributed Relational Database Architecture.

*System-directed access* enables one DB2 z/OS subsystem to execute SQL statements on another DB2 z/OS subsystem. System-directed access uses a private protocol only for DB2. It is known as a private protocol because you can use only it between DB2 databases. IBM recommends that users use DRDA. Although SAS/ACCESS Interface to DB2 under z/OS cannot explicitly request a connection, it can instead perform an implicit connection when SAS initiates a distributed request. To initiate an implicit connection, you must specify the **LOCATION=** option. When you specify this option, the three-level table name (*location.authid.table*) is used in the SQL statement that SAS/ACCESS Interface to DB2 under z/OS generates. When the SQL statement that contains the three-level table name is executed, an implicit connection is made to the remote DB2 subsystem. The primary authorization ID of the initiating process must be authorized to connect to the remote location.

*Distributed Relational Database Architecture* (DRDA) is a set of protocols that lets a user access distributed data. This lets SAS/ACCESS Interface to DB2 under z/OS access multiple remote tables at various locations. The tables can be distributed among multiple platforms, and both like and unlike platforms can communicate with one another. In a DRDA environment, DB2 acts as the client, server, or both.

To connect to a DRDA remote server or location, SAS/ACCESS Interface to DB2 under z/OS uses an explicit connection. To establish an explicit connection, SAS/ACCESS Interface to DB2 under z/OS first connects to the local DB2 subsystem through an **attachment facility** (CAF or RRSAF). It then issues an SQL CONNECT statement to connect from the local DB2 subsystem to the remote DRDA server before it accesses data. To initiate a connection to a DRDA remote server, you must specify the **SERVER= connection option**. When you specify this option, SAS uses a separate connection for each remote DRDA location.

---

## DB2 Attachment Facilities (CAF and RRSAF)

By default, SAS/ACCESS Interface to DB2 under z/OS uses the Call Attachment Facility (CAF) to make its connections to DB2. SAS supports multiple CAF connections for a SAS session. Therefore, for a SAS server, all clients can have their own connections to DB2; multiple clients no longer have to share one connection. However, because CAF does not support sign-on, each connection that the SAS server makes to DB2 has the z/OS authorization identifier of the server. It does not have the authorization identifier of the client for which the connection is made.

If you specify the **DB2ERRS** system option, SAS/ACCESS Interface to DB2 under z/OS engine uses the Recoverable Resource Manager Services Attachment Facility (RRSAF). Only one attachment facility can be used at a time, so the DB2RRS or NODB2RRS system option can be specified only when a SAS session is started. SAS supports multiple RRSAF connections for a SAS session. RRSAF was a new feature in DB2 Version 5, Release 1, and its support in SAS/ACCESS Interface to DB2 under z/OS was new in SAS 8.

The RRSAF is intended for use by SAS servers, such as the ones that SAS/SHARE software use. RRSAF supports the ability to associate a z/OS authorization identifier with each connection at sign-on. This authorization identifier is not the same as the authorization ID that you specify in the **AUTHID=** data set option or LIBNAME option. When connections use the System Authorization Facility and other security products, such as RACF, DB2 uses the RRSAF-supported authorization identifier to

validate a given connection's authorization to use both DB2 and system resources. This authorization identifier is basically the user ID with which you are logged on to z/OS.

With RRSAF, the SAS server makes the connections for each client. These connections are associated with the client z/OS authorization identifier. This is true only for clients that the SAS server authenticated, which occurred when the client specified a user ID and password. Servers authenticate their clients when the clients provide their user IDs and passwords. Generally, this is the default way that servers are run. If a client connects to a SAS server without providing a user ID and password, the identifier associated with its connections is that of the server (as with CAF) and not the client's identifier.

Other than specifying DB2RRS at SAS start-up, you do not need to do anything else to use RSSAF. the DB2 interface automatically signs on each connection that it makes to DB2 with the identifier of either the authenticated client or of the SAS server for non-authenticated clients. Authenticated clients have the same authorities to DB2 as they have when they run their own SAS session from their own ID and access DB2.

## Accessing DB2 System Catalogs

For many types of SAS procedures, the DB2 interface must access DB2 system catalogs for information. This information is limited to a list of all tables for a specific authorization identifier. The interface generates this SQL query to obtain information from system catalogs:

```
SELECT NAME FROM SYSIBM.SYSTABLES
  WHERE (CREATOR = 'authid');
```

Unless you specify the [AUTHID=](#) option, the authorization ID is the z/OS user ID that is associated with the job step.

The SAS procedures or applications that request the list of DB2 tables includes, but is not limited to, PROC DATASETS and PROC CONTENTS, or any application that needs a member list. If the SAS user does not have the necessary authorization to read the DB2 system catalogs, the procedure or application fails.

Because querying the DB2 system catalogs can cause some locking contentions, SAS/ACCESS Interface to DB2 under z/OS initiates a separate connection for the query to the DB2 system catalogs. After the query completes, a COMMIT WORK command is executed.

Under certain circumstances, you can access a catalog file by overriding the default value for the [DB2CATALOG=](#) system option.

---

# Sample Programs for DB2 under z/OS Hosts

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to Google BigQuery

---

|                                                                        |     |
|------------------------------------------------------------------------|-----|
| <i>System Requirements for SAS/ACCESS Interface to Google BigQuery</i> | 848 |
| <i>Introduction to SAS/ACCESS Interface to Google BigQuery</i>         | 848 |
| <i>LIBNAME Statement for the Google BigQuery Engine</i>                | 849 |
| Overview                                                               | 849 |
| Arguments                                                              | 849 |
| Google BigQuery LIBNAME Examples                                       | 852 |
| <i>Data Set Options for Google BigQuery</i>                            | 853 |
| <i>Connecting to Google BigQuery Using OAuth Authentication</i>        | 854 |
| Related LIBNAME Options                                                | 854 |
| Utility to Obtain an Authentication Token                              | 855 |
| <i>SQL Pass-Through Facility Specifics for Google BigQuery</i>         | 857 |
| Key Information                                                        | 857 |
| CONNECT Statement Example                                              | 857 |
| <i>Known Limitations When Working with Google BigQuery</i>             | 858 |
| Serial Operations                                                      | 858 |
| Table Aliases                                                          | 858 |
| Query Limitations for Google BigQuery                                  | 858 |
| MODIFY Statement Is Not Supported for Google BigQuery                  | 859 |
| <i>Working with Views for Google BigQuery</i>                          | 859 |
| <i>Updating and Deleting Google BigQuery Data</i>                      | 859 |
| <i>Passing SAS Functions to Google BigQuery</i>                        | 860 |
| <i>Passing Joins to Google BigQuery</i>                                | 862 |
| <i>Bulk Loading and Bulk Unloading for Google BigQuery</i>             | 862 |
| Options to Use with Bulk Loading and Unloading                         | 862 |
| Examples                                                               | 863 |
| Bulk Loading with PROC FEDSQL and PROC DS2                             | 863 |
| <i>Naming Conventions for Google BigQuery</i>                          | 863 |
| <i>Data Types and Conversions for Google BigQuery</i>                  | 864 |

|                                                    |     |
|----------------------------------------------------|-----|
| Overview .....                                     | 864 |
| Supported Google BigQuery Data Types .....         | 864 |
| LIBNAME Statement Data Conversions .....           | 865 |
| Working with Binary Data for Google BigQuery ..... | 867 |
| Working with ARRAY Data .....                      | 867 |

---

# System Requirements for SAS/ACCESS Interface to Google BigQuery

You can find information about system requirements for SAS/ACCESS Interface to Google BigQuery in the following locations:

- [System Requirements for SAS 9.4](#)
  - [SAS Viya System Requirements](#)
  - [Third-Party Software Requirements for Use with SAS 9.4](#)
- 

# Introduction to SAS/ACCESS Interface to Google BigQuery

For available SAS/ACCESS features and platforms, see [Google BigQuery supported features on page 101](#). For more information about Google BigQuery, see your Google BigQuery documentation.

SAS/ACCESS Interface to Google BigQuery includes SAS Data Connector to Google BigQuery. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- “Where to Specify Data Connector Options” in [SAS Cloud Analytic Services: User’s Guide](#)
- “Google BigQuery Data Connector” in [SAS Cloud Analytic Services: User’s Guide](#)

---

# LIBNAME Statement for the Google BigQuery Engine

---

## Overview

This section describes the LIBNAME statement options that SAS/ACCESS Interface to Google BigQuery supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Google BigQuery.

**LIBNAME** *libref* **bigquery** <connection-options> <LIBNAME-options>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

*bigquery*

specifies the SAS/ACCESS engine name for the Google BigQuery interface.

*connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the Google BigQuery database in several ways.

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

*CRED\_PATH='path-and-filename'*

specifies the location of a credential file that enables authentication to Google Cloud Platform.

This value is masked in the SAS log. This option accepts values that have been encoded using PROC PWENCODE. SAS/ACCESS recognizes encoded values as those that begin with a SAS encoding tag. For more information, see “[PWENCODE Procedure](#)” in [Base SAS Procedures Guide](#).

For more information about the credential file, see the getting started information in your Google Cloud documentation.

Valid in: SAS/ACCESS LIBNAME statement

Aliases: CREDPATH=, CRED\_FILE=, CREDFILE=

Default: none

Requirement: You must use single quotation marks around the value for CRED\_PATH=.

Example:

```
cred_path='/u/authfiles/BigQuery/xxx-yyy-8e99c10a22537.json'
```

**PROJECT='project-ID'**

specifies the project ID for a Google Cloud Platform project.

Valid in: SAS/ACCESS LIBNAME statement

Default: none

Requirement: This value is required to access Google BigQuery in a LIBNAME statement.

Example: project='project1'

#### *LIBNAME options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Google BigQuery with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

*Table 20.1 SAS/ACCESS LIBNAME Options for Google BigQuery*

| Option               | Default Value                                                            | Valid in<br>CONNECT |
|----------------------|--------------------------------------------------------------------------|---------------------|
| ACCESS=              | none                                                                     |                     |
| ALLOW_LARGE_RESULTS= | OFF                                                                      |                     |
| BL_BUCKET=           | none                                                                     | •                   |
| BL_DEFAULT_DIR=      | temporary file directory that is specified by the UTILLOC= system option | •                   |
| BL_DELETE_DATAFILE=  | YES                                                                      | •                   |
| BL_DELIMITER=        | bell character (ASCII 0x07)                                              | •                   |
| BL_NUM_READ_THREADS= | 4                                                                        | •                   |

| Option                         | Default Value                                                                   | Valid in CONNECT |
|--------------------------------|---------------------------------------------------------------------------------|------------------|
| BULKLOAD=                      | NO                                                                              | •                |
| BULKUNLOAD=                    | NO                                                                              |                  |
| CLIENT_ID=                     | none                                                                            |                  |
| CLIENT_SECRET=                 | none                                                                            |                  |
| DBCLIENT_MAX_BYTES=            | Maximum number of bytes per character for the current session encoding          | •                |
| DBCREATE_TABLE_OPTS=           | none                                                                            |                  |
| DBGEN_NAME=                    | DBMS                                                                            | •                |
| DBSASLABEL=                    | COMPAT                                                                          |                  |
| DEFER=                         | NO                                                                              | •                |
| DIRECT_SQL=                    | YES                                                                             |                  |
| DRIVER_TRACE=                  | none                                                                            |                  |
| DRIVER_TRACEFILE=              | none                                                                            |                  |
| DRIVER_TRACEOPTIONS=           | Trace file is overwritten and contains no time stamps or thread identification. |                  |
| INSERTBUFF=                    | automatically calculated based on row length                                    |                  |
| LARGE_RESULTS_DATASET=         | _sasbq_temp_tables                                                              |                  |
| LARGE_RESULTS_EXPIRATION_TIME= | 86400000                                                                        |                  |
| MAX_BINARY_LEN=                | 2000                                                                            |                  |
| MAX_CHAR_LEN=                  | 2000                                                                            |                  |
| MULTI_DATASRC_OPT=             | NONE                                                                            |                  |

| Option                  | Default Value                                | Valid in CONNECT |
|-------------------------|----------------------------------------------|------------------|
| PROXY=                  | none                                         |                  |
| QUALIFIER=              | none                                         |                  |
| READ_MODE=              | STANDARD                                     |                  |
| READBUFF=               | automatically calculated based on row length | •                |
| REFRESH_TOKEN=          | none                                         |                  |
| SCANSTRINGCOLUMNS=      | NO                                           |                  |
| SCHEMA=                 | none                                         |                  |
| SPOOL=                  | YES                                          |                  |
| SQL_FUNCTIONS=          | none                                         |                  |
| SQL_FUNCTIONS_COPY=     | none                                         |                  |
| SQLGENERATION=          | none                                         |                  |
| STRINGDATES=            | NO                                           | •                |
| TRACE=                  | NO                                           | •                |
| TRACEFILE=              | none                                         | •                |
| TRACEFLAGS=             | none                                         |                  |
| USE_INFORMATION_SCHEMA= | YES                                          |                  |

## Google BigQuery LIBNAME Examples

Use the following example to establish a connection between SAS and Google BigQuery. This example connects to project Project1 in schema Meddata.

```
libname mydb bigquery project='Project1' schema='Meddata';
```

If you require a credentials file, you might specify a LIBNAME statement like this one:

```
libname mydb bigquery project='Project1' schema='Meddata'
```

```
cred_path='/u/fedadmin/BigQuery/xxx-
yyy-9e99c10a8888.json';
```

---

# Data Set Options for Google BigQuery

All SAS/ACCESS data set options in this table are supported for Google BigQuery. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 20.2** SAS/ACCESS Data Set Options for Google BigQuery

| Option               | Default Value                                                            |
|----------------------|--------------------------------------------------------------------------|
| BL_BUCKET=           | none                                                                     |
| BL_DEFAULT_DIR=      | temporary file directory that is specified by the UTILLOC= system option |
| BL_DELETE_DATAFILE=  | YES                                                                      |
| BL_DELIMITER=        | the bell character (ASCII 0x07)                                          |
| BL_NUM_READ_THREADS= | LIBNAME option value                                                     |
| BULKLOAD=            | LIBNAME option value                                                     |
| BULKUNLOAD=          | LIBNAME option value                                                     |
| DBCONDITION=         | none                                                                     |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                                                     |
| DBFORCE=             | NO                                                                       |
| DBGEN_NAME=          | DBMS                                                                     |
| DBLABEL=             | NO                                                                       |
| DBLARGETABLE=        | NO                                                                       |
| DBNULL=              | none                                                                     |
| DBSASTYPE=           | see “ <a href="#">Data Types and Conversions for Google BigQuery</a> ”   |
| DBTYPE=              | see “ <a href="#">Data Types and Conversions for Google BigQuery</a> ”   |

| Option             | Default Value                                        |
|--------------------|------------------------------------------------------|
| ERRLIMIT=          | 1                                                    |
| INSERTBUFF=        | LIBNAME option value                                 |
| NULLCHAR=          | SAS                                                  |
| NULLCHARVAL=       | a blank character                                    |
| QUALIFIER=         | LIBNAME option value                                 |
| READ_MODE=         | LIBNAME option value                                 |
| READBUFF=          | LIBNAME option value                                 |
| SASDATEFMT=        | see “Data Types and Conversions for Google BigQuery” |
| SCANSTRINGCOLUMNS= | LIBNAME option value                                 |
| SCHEMA=            | LIBNAME option value                                 |

---

## Connecting to Google BigQuery Using OAuth Authentication

---

### Related LIBNAME Options

Use these options to connect to Google BigQuery using the OAuth tool for authentication:

- CLIENT\_ID=
- CLIENT\_SECRET=
- REFRESH\_TOKEN=

Here is an example of a LIBNAME statement that connects to Google BigQuery using the OAuth tool options:

```
libname mylib bigquery schema='Region1'
      project='myProject'
      CLIENT_ID="919191919191-abababa5ab96so8v9o298np4tg0la1md.apps.googleusercontent.com"
      CLIENT_SECRET=O8XKYBEdFZdX_1BAbABab9AB
      REFRESH_TOKEN="1//0dJj ... 13-7c0oDxCR0p9u6k3_jl_i-Y31hdZ4FOumSO0"
```

```
<other-options>;
```

Here is a sample call to PROC SQL that uses the same options:

```
proc sql;
  connect to (CLIENT_SECRET=08XKYBEDFZdX_1BAbABab9AB
    CLIENT_ID="919191919191-abababa5ab96so8v9o298np4tg0la1md.apps.googleusercontent.com"
    REFRESH_TOKEN="1//0dJj ... 13-7c0oDxCR0p9u6k3_jl_i-Y31hdZ4FOumSO0"
    <other-options>);
```

---

**Note:** The value of REFRESH\_TOKEN= is typically quite long. The ellipsis (...) symbol is used to convey that the value is longer than what is shown in the examples.

---

## Utility to Obtain an Authentication Token

### Install the Authentication Token Utility

SAS/ACCESS Interface to Google BigQuery provides a utility that generates the refresh token that is needed for OAuth authentication. This utility is part of the SAS/ACCESS to Google BigQuery Tools that are available at <https://support.sas.com/downloads/package.htm?pid=2459>.

The name of the file to download corresponds to your operating system.

- Linux: sas-bigquery-tools-<version>-linux.tgz
- macOS: sas-bigquery-tools-<version>-osx.tgz
- Windows: sas-bigquery-tools-<version>.zip

The Authentication Token utility is called getRefreshToken, and it is contained in the compressed file.

### Run the Authentication Token Utility

To obtain a refresh token using the Authentication Token utility for the Google BigQuery interface:

- 1 Run the getRefreshToken utility, providing your client ID and client secret values. The client\_id is the ID for the OAuth client for which you are obtaining a refresh token. The client\_secret is the secret value for the OAuth client.

```
utility-path/getRefreshToken -client_id client_id
  -client_secret client_secret
```

- 2 Open a web browser and access the URL that you obtained in step 1.
  - a Provide your Google credentials to log in to the site.
  - b Click **Allow** to grant Google BigQuery and GCP services access to your data.

- c You receive an error that states that you cannot access the site. However, you can copy the necessary access token from the resulting URL. Copy the value after the `&code` parameter (the value ends before the `&scope` parameter). This is your access token.
- 3 To obtain a refresh token, enter the access token value that you obtained in step 2.

To test your connection with a refresh token value, you can enter a command similar to the following:

```
utility-path/getRefreshToken -client_id client-ID
    -client_secret secret
    -project myProject
    -token refresh-token-value
    -query 'select * from `schema.table`'
```

To get help with syntax for `getRefreshToken`, specify the following command:

```
getRefreshToken -h
```

The `getRefreshToken` command supports these options:

|                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-client_id</code> <i>string</i>     | specifies the OAuth client ID value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>-client_secret</code> <i>string</i> | specifies the OAuth client secret.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>-project</code> <i>string</i>       | specifies the project to connect with for your test query.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>-proxy</code> <i>string</i>         | specifies the proxy URL to connect through.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>-query</code> <i>string</i>         | specifies a test query to run to test your connection with the refresh token. Specify the <code>-project</code> option when you specify <code>-query</code> . Enclose the query in single quotation marks ('') and enclose table values with back quotation marks (`'). For example, you might specify this test query: <code>-query 'select * from `schema.table`'</code>                                                                                                                                                                                                |
| <code>-scope</code> <i>string</i>         | specifies the scope for the token. This can take the form of a keyword, a combination of keywords, or one or more scope URLs. Valid keywords are <code>bigrquery</code> , <code>sheets</code> , <code>drive</code> , or <code>all</code> . Use a <code>+</code> to combine keywords, such as <code>sheets+drive</code> . The keyword <code>all</code> must be specified by itself. Use a <code>+</code> to combine multiple scope URL values, such as <code>https://www.googleapis.com/auth/drive.readonly+https://www.googleapis.com/auth/spreadsheets.readonly</code> . |
| <code>-token</code> <i>string</i>         | specifies the refresh token to connect with.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

---

# SQL Pass-Through Facility Specifics for Google BigQuery

---

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Google BigQuery interface.

- The *dbms-name* is **bq**.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Google BigQuery. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default **bq** alias is used.
- The CONNECT statement’s *database-connection-arguments* are identical to its LIBNAME [connection options](#).

---

## CONNECT Statement Example

This example uses the DBCON alias to connect to **mysrv1** the Google BigQuery database and execute a query. The connection alias is optional.

```
proc sql;
  connect to bq as dbcon
    (cred_path='/u/BigQuery/xxx-yyy-9e99c10a8888.json'
     project='project1' schema='Region1');
  select * from connection to dbcon
    (select * from customers WHERE customer like '1%');
  quit;
```

---

# Known Limitations When Working with Google BigQuery

---

## Serial Operations

The streaming nature of Google BigQuery presents challenges for operations that occur serially. These operations include the following actions:

- creating a table and then inserting columns
- dropping a table and then re-creating it
- updating or deleting rows that were recently added

If the data feed for the second operation begins before the first operation has completed, data can be lost. To avoid these streaming issues, enable bulk loading using SAS/ACCESS bulk-load options. For more information, see “[Bulk Loading and Bulk Unloading for Google BigQuery](#)” on page 862.

---

## Table Aliases

Do not use table aliases in PROC SQL code that you pass to Google BigQuery when you invoke the UPDATE or DELETE statements.

---

**Note:** The DBIDIRECTEXEC system option must be enabled to use the UPDATE or DELETE statements with Google BigQuery. The DBIDIRECTEXEC system option is enabled by default for Google BigQuery.

---

---

## Query Limitations for Google BigQuery

If you run a job that queries the Google BigQuery database, you might receive an error similar to the following:

```
ERROR: Error retrieving table list: Error: googleapi: Error 400: Job exceeded  
rate limits: Your project_and_region exceeded quota for concurrent  
queries. For more information, see  
https://cloud.google.com/bigquery/troubleshooting-errors,  
jobRateLimitExceeded
```

Google BigQuery limits the number of incoming query requests. For more information, see <https://cloud.google.com/bigquery/quotas>.

---

## MODIFY Statement Is Not Supported for Google BigQuery

SAS/ACCESS Interface to Google BigQuery does not support positional inserts, updates, or deletions. Therefore, the MODIFY statement in a DATA step is not supported for Google BigQuery.

---

## Working with Views for Google BigQuery

When the DBIDIRECTEXEC system option is enabled (default), any time that you call the CREATE VIEW clause in SQL code, a FedSQL view is created. A FedSQL view is stored in a DBMS table that has the same name as the FedSQL view. Although you can access the data in a FedSQL view from within SAS, the data in the DBMS table that represents the FedSQL view cannot be read by any other tool.

To generate a DBMS view that can be read, use explicit SQL pass-through with PROC SQL by using Google BigQuery syntax. An explicit SQL pass-through connection uses the CONNECT statement from PROC SQL. For more information, see [Explicit SQL Pass-Through](#).

---

## Updating and Deleting Google BigQuery Data

To update or delete content, you must use PROC SQL to submit code to the Google BigQuery database. You must also ensure that the DBIDIRECTEXEC system option is enabled. DBIDIRECTEXEC is enabled by default for Google BigQuery.

The following example updates the value of the variable Price in table Stock23 when the variable Instock is greater than 50.

```
libname x1 bigquery project='Project1'  
      schema='Region1';  
  
proc sql;  
  update x1.stock23 set price=price*1.1  
    where instock > 50;  
quit;
```

The following example deletes rows from table Mydata where the variable A is set to 1.

```

libname x1 bigquery project='Project1'
      schema='Region1';

proc sql;
  delete from x1.mydata where a=1;
quit;

```

This example deletes any row where the variable A is not specified (is missing). The database checks for null values.

```

libname x1 bigquery project='Project1'
      schema='Region1';

proc sql;
  delete from x1.mydata where a=.;
quit;

```

## Passing SAS Functions to Google BigQuery

SAS/ACCESS Interface to Google BigQuery passes the following SAS functions to Google BigQuery for processing. When the Google BigQuery function name differs from the SAS function name, the Google BigQuery name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on [page 58](#).

|                |                |
|----------------|----------------|
| ABS            | LOG10          |
| ARCOS (ACOS)   | LOWCASE        |
| ARSIN (ASIN)   | MAX            |
| ATAN           | MIN            |
| ATAN2          | SIGN           |
| AVG            | SIN            |
| CEILING (CEIL) | SQRT           |
| COS            | STDDEV         |
| COUNT          | SUM            |
| EXP            | TAN            |
| FLOOR          | UPCASE         |
| LOG            | VAR (VARIANCE) |

`SQL_FUNCTIONS=ALL` allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when `SQL_FUNCTIONS=ALL` can the SAS/ACCESS engine also pass these SAS SQL functions to Google BigQuery. Because of incompatibility in date and time functions between Google BigQuery and SAS, Google BigQuery might not process them correctly. Check your results to determine whether these functions are working as expected. For more information, see “[SQL\\_FUNCTIONS=LIBNAME Statement Option](#)” on [page 324](#).

**Table 20.3** Differences in Function Behavior between SAS and Google BigQuery

| SAS Function | Google BigQuery Function  | Usage Notes                                                                                                                     |
|--------------|---------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| DATE         | CURRENT_DATE              | Value is passed as a constant only when the DBMS uses the CURRENT_DATE function.                                                |
| DATETIME     | CURRENT_TIMESTAMP         | Value is passed as a constant only when the DBMS uses the CURRENT_TIMESTAMP function.                                           |
| HOUR         | EXTRACT (HOUR FROM ...)   | SAS passes this function through for DATETIME, TIME, and TIMESTAMP values. This function is not passed through for DATE values. |
| LENGTH       | BYTE_LENGTH               | <i>none</i>                                                                                                                     |
| MINUTE       | EXTRACT (MINUTE FROM ...) | SAS passes this function through for DATETIME, TIME, and TIMESTAMP values. This function is not passed through for DATE values. |
| MOD          | MOD                       | The MOD function does not pass FLOAT64 values, because SAS does not modify non-integer arguments to the MOD function.           |
| MONTH        | EXTRACT (MONTH FROM ...)  | SAS passes this function through for DATE, DATETIME, and TIMESTAMP values. This function is not passed through for TIME values. |
| SECOND       | EXTRACT (SECOND FROM ...) | SAS passes this function through for DATETIME, TIME, and TIMESTAMP values. This function is not passed through for DATE values. |
| TIME         | CURRENT_TIME              | Value is passed as a constant only when the DBMS uses the CURRENT_TIME function.                                                |
| TRIM         | RTRIM                     | SAS and Google BigQuery handle empty strings differently.                                                                       |

| SAS Function | Google BigQuery Function | Usage Notes                                                                                                                     |
|--------------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| TODAY        | CURRENT_DATE             | Value is passed as a constant only when the DBMS uses the CURRENT_DATE function.                                                |
| YEAR         | EXTRACT (YEAR FROM ...)  | SAS passes this function through for DATE, DATETIME, and TIMESTAMP values. This function is not passed through for TIME values. |

## Passing Joins to Google BigQuery

In order for a multiple-libref join to pass to Google BigQuery, the CRED\_PATH= value in the LIBNAME statements must match exactly. For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

## Bulk Loading and Bulk Unloading for Google BigQuery

### Options to Use with Bulk Loading and Unloading

Bulk loading is the fastest way to insert large numbers of rows into a Google BigQuery table. You must enable either the **BULKLOAD= LIBNAME** option or the **BULKLOAD=** data set option in order to use the bulk-load interface. The Google BigQuery bulk-load interface moves data from SAS to the Google BigQuery database.

Similarly, the fastest way to retrieve a large number of rows from Google BigQuery is to use bulk unloading. You must enable either the **BULKUNLOAD= LIBNAME** option or the **BULKUNLOAD=** data set option to use the Google BigQuery Extractor API.

**Note:**

Bulk loading record data is not supported. Bulk unloading record data is supported.

Here are the Google BigQuery bulk-load and bulk-unload options.

- BL\_BUCKET= [LIBNAME option](#) and [data set option](#)
- BL\_DEFAULT\_DIR= [LIBNAME option](#) and [data set option](#)
- BL\_DELETE\_DATAFILE= [LIBNAME option](#) and [data set option](#)
- BL\_DELIMITER= [LIBNAME option](#) and [data set option](#)
- BL\_NUM\_READ\_THREADS= [LIBNAME option](#) and [data set option](#)
- BULKLOAD= [LIBNAME option](#) and [data set option](#)
- BULKUNLOAD= “[BULKUNLOAD= LIBNAME Statement Option](#)” [LIBNAME option](#) and “[BULKUNLOAD= Data Set Option](#)” [data set option](#)

## Examples

This example shows how you can use a SAS data set, Sasflt.Flt98, to create and load a large Google BigQuery table, net\_air.flights98.

```
libname sasflt 'SAS-library';
libname net_air bigquery project='Project1' schema='USdata'
      cred_path='/u/fedadmin/BigQuery/xxx-yyy-9e99c10a8888.json';

proc sql;
create table net_air.flights98(bulkload=YES)
      as select * from sasflt.flt98;
quit;
```

## Bulk Loading with PROC FEDSQL and PROC DS2

Any bulk-load options that you specify in a SAS/ACCESS LIBNAME statement are automatically included in the connection string that is passed from PROC FEDSQL or PROC DS2. You can override these values by specifying table options within these procedures.

## Naming Conventions for Google BigQuery

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Google BigQuery interface supports table names up to 1024 characters and column names up to 128 characters. If column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical column names,

SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or fewer. SAS does not truncate a name longer than 32 characters. If you have a table name that is longer than 32 characters, it is recommended that you create a table view. Google BigQuery is not case sensitive, so all names default to lowercase.

Google BigQuery objects include tables, views, and columns. They follow these conventions.

- SAS supports table or column names up to 32 characters.
- A name must begin with a letter (A through Z) or an underscore (\_).
- To enable case sensitivity, enclose names in quotation marks.
- A name cannot be a reserved word in Google BigQuery such as WHERE or VIEW.
- A name cannot be the same as another Google BigQuery object that has the same type.

---

## Data Types and Conversions for Google BigQuery

---

### Overview

Every column in a table has a name and a data type. The data type tells Google BigQuery how much physical storage to set aside for the column and the form in which the data is stored.

For information about Google BigQuery data types and to determine which data types are available for your version of Google BigQuery, see your Google BigQuery documentation.

---

## Supported Google BigQuery Data Types

---

Here are the data types that the Google BigQuery engine supports.

- Character data:

BYTES      STRING

- Numeric data:

BOOL      INT64  
FLOAT64    NUMERIC

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Date, time, and timestamp data:

|          |           |
|----------|-----------|
| DATE     | TIME      |
| DATETIME | TIMESTAMP |

**Note:** Be aware that columns of these data types can contain data values that are out of range for SAS.

- Other data types:

|           |                                     |
|-----------|-------------------------------------|
| ARRAY     | RECORD (also referred to as STRUCT) |
| GEOGRAPHY |                                     |

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Google BigQuery assigns to SAS variables when using the [LIBNAME statement](#) to read from a Google BigQuery table. These default formats are based on Google BigQuery column attributes.

**Table 20.4 LIBNAME Statement: Default SAS Formats for Google BigQuery Data Types**

| Google BigQuery Data Type | SAS Data Type | Default SAS Format |
|---------------------------|---------------|--------------------|
| <i>Character Data</i>     |               |                    |
| BYTES                     | CHAR          | \$HEXw.            |
| STRING                    | CHAR          | \$w.               |
| <i>Numeric Data</i>       |               |                    |
| BIGINT                    | NUMERIC       | 20.                |
| BOOL                      | NUMERIC       | 1.                 |
| FLOAT64                   | NUMERIC       | none               |

| <b>Google BigQuery Data Type</b> | <b>SAS Data Type</b> | <b>Default SAS Format</b> |
|----------------------------------|----------------------|---------------------------|
| NUMERIC                          | NUMERIC              | none                      |
| <i>Date and Time Data</i>        |                      |                           |
| DATE                             | NUMERIC              | DATE9.                    |
| DATETIME                         | NUMERIC              | DATETIME25.6              |
| TIME                             | NUMERIC              | TIME15.6                  |
| TIMESTAMP                        | NUMERIC              | DATETIME25.6              |
| Other Data                       |                      |                           |
| ARRAY                            | VARCHAR              | \$w. <sup>2</sup>         |
| GEOGRAPHY                        | VARCHAR              | \$w. <sup>2</sup>         |
| RECORD <sup>1</sup>              | VARCHAR              | \$w. <sup>2</sup>         |

<sup>1</sup> Also referred to as a STRUCT in Google BigQuery.

<sup>2</sup> The default length is based on MAX\_CHAR\_LEN.

This table shows the default Google BigQuery data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 20.5 LIBNAME Statement: Default Google BigQuery Data Types for SAS Variable Formats**

| <b>SAS Variable Format</b> | <b>Google BigQuery Data Type</b> |
|----------------------------|----------------------------------|
| w.d                        | DOUBLE                           |
| w.                         | BIGINT                           |
| \$w.                       | VARCHAR                          |
| datetime formats           | DATETIME                         |
| date formats               | DATE                             |
| time formats               | TIME                             |

---

## Working with Binary Data for Google BigQuery

SAS/ACCESS Interface to Google BigQuery supports the use of BINARY data. When binary data is displayed, the value is typically padded with spaces (0x20) at the end of a value. To control the length of values that are displayed when using PROC PRINT, PROC SQL, or other procedures, specify a value for the **MAX\_BINARY\_LEN= LIBNAME** option. Otherwise, these procedures might truncate the data with different lengths.

---

## Working with ARRAY Data

When you load ARRAY data into SAS, the data is loaded as VARCHAR data. The default length of the data is based on MAX\_CHAR\_LEN.

To work with ARRAY data as the data type that is assigned in Google BigQuery, use PROC SQL to pass SQL queries down to the database.



# SAS/ACCESS Interface to Greenplum

---

|                                                                        |     |
|------------------------------------------------------------------------|-----|
| <i>System Requirements for SAS/ACCESS Interface to Greenplum</i> ..... | 870 |
| <i>Introduction to SAS/ACCESS Interface to Greenplum</i> .....         | 870 |
| <b>LIBNAME Statement for the Greenplum Engine</b> .....                | 871 |
| Overview .....                                                         | 871 |
| Arguments .....                                                        | 871 |
| Greenplum LIBNAME Statement Examples .....                             | 875 |
| <b>Data Set Options for Greenplum</b> .....                            | 875 |
| <b>SQL Pass-Through Facility Specifics for Greenplum</b> .....         | 878 |
| Key Information .....                                                  | 878 |
| CONNECT Statement Example .....                                        | 879 |
| Special Catalog Queries .....                                          | 879 |
| <b>Autopartitioning Scheme for Greenplum</b> .....                     | 880 |
| Overview .....                                                         | 880 |
| Autopartitioning Restrictions .....                                    | 880 |
| Nullable Columns .....                                                 | 881 |
| Using WHERE Clauses .....                                              | 881 |
| Using DBSliceparm= .....                                               | 881 |
| Using DBslice= .....                                                   | 881 |
| <b>Temporary Table Support for Greenplum</b> .....                     | 882 |
| <b>Passing SAS Functions to Greenplum</b> .....                        | 883 |
| <b>Passing Joins to Greenplum</b> .....                                | 884 |
| <b>Sorting Data That Contains NULL Values</b> .....                    | 884 |
| <b>Bulk Loading for Greenplum</b> .....                                | 885 |
| Overview .....                                                         | 885 |
| Using Protocols to Access External Tables .....                        | 885 |
| Configuring the File Server .....                                      | 886 |
| Stopping gpfdist .....                                                 | 886 |
| Troubleshooting gpfdist .....                                          | 887 |
| Using the file:// Protocol .....                                       | 887 |

|                                                 |     |
|-------------------------------------------------|-----|
| Accessing Dynamic Data in Web Tables .....      | 887 |
| Data Set Options for Bulk Loading .....         | 887 |
| Bulk Loading Examples .....                     | 888 |
| <i>Locking in the Greenplum Interface</i> ..... | 889 |
| <i>Naming Conventions for Greenplum</i> .....   | 890 |
| <i>Data Types for Greenplum</i> .....           | 891 |
| Overview .....                                  | 891 |
| Supported Greenplum Data Types .....            | 891 |
| Greenplum Null Values .....                     | 892 |
| LIBNAME Statement Data Conversions .....        | 892 |
| <i>Sample Programs for Greenplum</i> .....      | 894 |

---

# System Requirements for SAS/ACCESS Interface to Greenplum

You can find information about system requirements for SAS/ACCESS Interface to Greenplum in the following locations:

- [System Requirements for SAS/ACCESS Interface to Greenplum with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to Greenplum

For available SAS/ACCESS features, see [Greenplum supported features on page 102](#). For more information about Greenplum, see your Greenplum documentation.

---

# LIBNAME Statement for the Greenplum Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Greenplum supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Greenplum.

**LIBNAME** *libref* **greenplm** <*connection-options*> <*LIBNAME-options*>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

*greenplm*

specifies the SAS/ACCESS engine name for the Greenplum interface.

*connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the Greenplum database in two ways. Specify only one of these methods for each connection because they are mutually exclusive.

- SERVER=, DATABASE=, PORT=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**SERVER=<'>*server-name*<'>**

specifies the Greenplum server name or the IP address of the server host. If the server name contains spaces or nonalphanumeric characters or if it is an IP address, you must enclose it in quotation marks.

Alias: HOST=

**DATABASE=<'>*database-name*<'**

specifies the Greenplum database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORT=*port***

specifies the port number that is used to connect to the specified Greenplum database. If you do not specify a port, the default is 5432.

**USER=<'>*Greenplum-user-name*<'**

specifies the Greenplum user name (also called the user ID) that is used to connect to the database. If the user name contains spaces or nonalphanumeric characters, use quotation marks.

Alias: UID=

**PASSWORD=<'>*Greenplum-password*<'**

specifies the password that is associated with your Greenplum user ID. If the password contains spaces or nonalphabetic characters, you must enclose it in quotation marks. You can also specify PASSWORD= with the PWD=, PASS=, and PW= aliases.

**DSN=<'>*Greenplum-data-source*<'**

specifies the configured Greenplum ODBC data source to which you want to connect. It is recommended that you use this option only if you have configured Greenplum ODBC data sources on your client. This method requires additional setup—either through the ODBC Administrator control panel on Windows platforms, or through the odbc.ini file or a similarly named configuration file on UNIX platforms. It is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

Aliases: DS=, DATASRC=

**LIBNAME -options**

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Greenplum with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 21.1** SAS/ACCESS LIBNAME Options for Greenplum

| Option      | Default Value      | Valid in CONNECT |
|-------------|--------------------|------------------|
| ACCESS=     | none               |                  |
| AUTHDOMAIN= | none               |                  |
| AUTOCOMMIT= | operation-specific | •                |
| BULKLOAD=   | NO                 | •                |

| Option                    | Default Value                                      | Valid in CONNECT |
|---------------------------|----------------------------------------------------|------------------|
| CONNECTION=               | SHAREDREAD                                         | •                |
| CONNECTION_GROUP=         | none                                               | •                |
| CONOPTS=                  | none                                               | •                |
| CURSOR_TYPE=              | FORWARD_ONLY                                       | •                |
| DBCOMMIT=                 | 1000 (when inserting rows), 0 (when updating rows) |                  |
| DBCONINIT=                | none                                               | •                |
| DBCONTERM=                | none                                               | •                |
| DBCREATE_TABLE_OPTS=      | none                                               |                  |
| DBGEN_NAME=               | DBMS                                               | •                |
| DBINDEX=                  | NO                                                 |                  |
| DBLIBINIT=                | none                                               |                  |
| DBLIBTERM=                | none                                               |                  |
| DBMAX_TEXT=               | 1024                                               | •                |
| DBMSTEMP=                 | NO                                                 |                  |
| DBNULLKEYS=               | YES                                                |                  |
| DBPROMPT=                 | NO                                                 | •                |
| DBSASLABEL=               | COMPAT                                             |                  |
| DBSLICEPARM=              | NONE                                               |                  |
| DEFER=                    | NO                                                 | •                |
| DELETE_MULT_ROWS=         | NO                                                 |                  |
| DIRECT_EXE=               | none                                               |                  |
| DIRECT_SQL=               | YES                                                |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                 |                  |

| Option                | Default Value                                      | Valid in CONNECT |
|-----------------------|----------------------------------------------------|------------------|
| IN=                   | none                                               |                  |
| INSERT_SQL=           | YES                                                |                  |
| INSERTBUFF=           | automatically calculated based on row length       |                  |
| KEYSET_SIZE=          | 0                                                  | •                |
| MULTI_DATASRC_OPT=    | NONE                                               |                  |
| POST_STMT_OPTS=       | none                                               |                  |
| PRESERVE_COL_NAMES=   | YES<br><br>See “Naming Conventions for Greenplum”. |                  |
| PRESERVE_TAB_NAMES=   | see “Naming Conventions for Greenplum”             |                  |
| QUERY_TIMEOUT=        | 0                                                  | •                |
| QUOTE_CHAR=           | none                                               |                  |
| READ_ISOLATION_LEVEL= | RC                                                 | •                |
| READ_LOCK_TYPE=       | ROW                                                | •                |
| READBUFF=             | automatically calculated based on row length       | •                |
| REREAD_EXPOSURE=      | NO                                                 | •                |
| SCHEMA=               | none                                               |                  |
| SPOOL=                | YES                                                |                  |
| SQL_FUNCTIONS=        | none                                               |                  |
| SQL_FUNCTIONS_COPY=   | none                                               |                  |
| SQLGENERATION=        | none                                               |                  |
| STRINGDATES=          | NO                                                 | •                |
| SUB_CHAR=             | none                                               |                  |

| Option                  | Default Value | Valid in CONNECT |
|-------------------------|---------------|------------------|
| TRACE=                  | none          | •                |
| TRACEFILE=              | none          | •                |
| UPDATE_ISOLATION_LEVEL= | RC            |                  |
| UPDATE_LOCK_TYPE=       | ROW           | •                |
| UPDATE_SQL=             | YES           |                  |
| UPDATE_MULT_ROWS=       | NO            |                  |
| UTILCONN_TRANSIENT=     | NO            |                  |

## Greenplum LIBNAME Statement Examples

In this example, SERVER=, DATABASE=, PORT=, USER=, and PASSWORD= are the connection options.

```
libname mydblib greenplm server=gplum04 db=customers port=5432
      user=gpusrl password=gppwd1;

proc print data=mydblib.customers;
  where state='CA';
run;
```

In the next example, DSN=, USER=, and PASSWORD= are the connection options. The Greenplum data source is configured in the ODBC Administrator Control Panel on Windows platforms. It is also configured in the odbc.ini file or a similarly named configuration file on UNIX platforms.

```
libname mydblib greenplm DSN=gplumSalesDiv user=gpusrl password=gppwd1;

proc print data=mydblib.customers;
  where state='CA';
run;
```

## Data Set Options for Greenplum

All SAS/ACCESS data set options in this table are supported for Greenplum. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 21.2** SAS/ACCESS Data Set Options for Greenplum

| Option               | Default Value                                    |
|----------------------|--------------------------------------------------|
| BL_DATAFILE_EXISTS=  | NO                                               |
| BL_DATAFILE=         | none                                             |
| BL_DELETE_DATAFILE=  | none                                             |
| BL_DELIMITER=        |                                                  |
| BL_ENCODING=         | DEFAULT                                          |
| BL_ESCAPE=           | \                                                |
| BL_EXCEPTION=        | none                                             |
| BL_EXECUTE_CMD=      | none                                             |
| BL_EXECUTE_LOCATION= | none                                             |
| BL_EXTERNAL_WEB=     | NO                                               |
| BL_FORCE_NOT_NULL=   | none                                             |
| BL_FORMAT=           | TEXT                                             |
| BL_HEADER=           | NO                                               |
| BL_HOST=             | 127.0.0.1                                        |
| BL_NULL=             | 'N' [TEXT mode], unquoted empty value [CSV mode] |
| BL_PORT=             | 8080                                             |
| BL_PROTOCOL=         | 'gpfdist'                                        |
| BL_QUOTE=            | " (double quotation mark)                        |
| BL_REJECT_LIMIT=     | none                                             |
| BL_REJECT_TYPE=      | ROWS                                             |
| BL_USE_PIPE=         | NO                                               |
| BULKLOAD=            | none                                             |
| DBCOMMIT=            | LIBNAME option value                             |

| Option                    | Default Value                  |
|---------------------------|--------------------------------|
| DBCONDITION=              | none                           |
| DBCREATE_TABLE_OPTS=      | LIBNAME option value           |
| DBFORCE=                  | none                           |
| DBGEN_NAME=               | DBMS                           |
| DBINDEX=                  | NO                             |
| DBKEY=                    | none                           |
| DBLABEL=                  | none                           |
| DBLARGETABLE=             | none                           |
| DBMAX_TEXT=               | 1024                           |
| DBNULL=                   | none                           |
| DBNULLKEYS=               | LIBNAME option value           |
| DBPROMPT=                 | LIBNAME option value           |
| DBSASTYPE=                | see “Data Types for Greenplum” |
| DBSLICE=                  | none                           |
| DBSLICEPARAM=             | NONE                           |
| DBTYPE=                   | see “Data Types for Greenplum” |
| DISTRIBUTED_BY=           | DISTRIBUTED_RANDOMLY           |
| ERRLIMIT=                 | 1                              |
| IGNORE_READ_ONLY_COLUMNS= | none                           |
| INSERTBUFF=               | LIBNAME option value           |
| KEYSET_SIZE=              | LIBNAME option value           |
| NULLCHAR=                 | SAS                            |
| NULLCHARVAL=              | a blank character              |
| POST_STMT_OPTS=           | none                           |

| Option                  | Default Value        |
|-------------------------|----------------------|
| POST_TABLE_OPTS=        | none                 |
| PRE_STMT_OPTS=          | none                 |
| PRE_TABLE_OPTS=         | none                 |
| PRESERVE_COL_NAMES=     | LIBNAME option value |
| QUERY_TIMEOUT=          | LIBNAME option value |
| READ_ISOLATION_LEVEL=   | LIBNAME option value |
| READ_LOCK_TYPE=         | LIBNAME option value |
| READBUFF=               | LIBNAME option value |
| SASDATEFMT=             | none                 |
| SCHEMA=                 | LIBNAME option value |
| SUB_CHAR=               | none                 |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |
| UPDATE_LOCK_TYPE=       | LIBNAME option value |

---

## SQL Pass-Through Facility Specifics for Greenplum

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Greenplum interface.

- The *dbms-name* is GREENPLM.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Greenplum. If you use multiple simultaneous connections, you must use the *alias* argument to identify the

different connections. If you do not specify an alias, the default GREENPLM alias is used.

- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME connection options.

## CONNECT Statement Example

This example uses the DBCON alias to connect to the greenplum04 Greenplum server database and execute a query. The connection alias is optional.

```
proc sql;
  connect to greenplm as dbcon
    (server=greenplum04 db=sample port=5432 user=gpusrl
     password=gppwd1);
  select * from connection to dbcon
    (select * from customers where customer like '1%');
quit;
```

## Special Catalog Queries

SAS/ACCESS Interface to Greenplum supports the following special queries. You can use the queries to call functions in ODBC-style function application programming interfaces (APIs). Here is the general format of the special queries:

`Greenplum::SQLAPI 'parameter-1', 'parameter-n'`

`Greenplum::`

is required to distinguish special queries from regular queries. `Greenplum::` is not case sensitive.

`SQLAPI`

is the specific API that is being called. `SQLAPI` is not case sensitive.

`'parameter n'`

is a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, this call to SQLTables usually matches table names such as myatest and my\_test:

```
select * from connection to greenplm
  (Greenplum::SQLTables "test","","my_test");
```

Use the escape character to search only for the my\_test table:

```
select * from connection to greenplm
  (Greenplum::SQLTables "test","","my\_test");
```

SAS/ACCESS Interface to Greenplum supports these special queries.

- Greenplum::SQLTables <'Catalog', 'Schema', 'Table-name', 'Type'>  
 returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.
- Greenplum::SQLColumns <'Catalog', 'Schema', 'Table-name', 'Column-name'>  
 returns a list of all columns that match the specified arguments. If you do not specify any arguments, all accessible column names and information are returned.
- Greenplum::SQLColumns <'Catalog', 'Schema', 'Table-name', 'Column-name'>  
 returns a list of all columns that match the specified arguments. If you do not specify any argument, all accessible column names and information are returned.
- Greenplum::SQLPrimaryKeys <'Catalog', 'Schema', 'Table-name' 'Type'>  
 returns a list of all columns that compose the primary key that matches the specified table. A primary key can be composed of one or more columns. If you do not specify any table name, this special query fails.
- Greenplum::SQLStatistics <'Catalog', 'Schema', 'Table-name'>  
 returns a list of the statistics for the specified table name. You can specify the SQL\_INDEX\_ALL and SQL\_ENSURE options in the SQLStatistics API call. If you do not specify any table name argument, this special query fails.
- Greenplum::SQLGetTypeInfo  
 returns information about the data types that the Greenplum database supports.

## Autopartitioning Scheme for Greenplum

### Overview

Autopartitioning for SAS/ACCESS Interface to Greenplum is a modulo (MOD) function method. For general information about this feature, see [“Autopartitioning Techniques in SAS/ACCESS” on page 76](#).

### Autopartitioning Restrictions

SAS/ACCESS Interface to Greenplum places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- INTEGER and SMALLINT columns are given preference.
- You can use other numeric columns for partitioning if the precision minus the scale of the column is greater than 0 but less than 10—namely,  $0 < (\text{precision} - \text{scale}) < 10$ .

---

## Nullable Columns

If you select a nullable column for autopartitioning, the `OR<column-name>IS NULL` SQL statement is appended at the end of the SQL code that is generated for the threaded Read. This ensures that any possible NULL values are returned in the result set.

---

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in a SAS WHERE clause. For example, this DATA step cannot use a threaded Read to retrieve the data because all numeric columns in the table are in the WHERE clause:

```
data work.locemp;
  set trlib.MYEMPS;
  where EMPNUM<=30 and ISTENURE=0 and
        SALARY<=35000 and NUMCLASS>2;
run;
```

---

## Using DBSLICEPARM=

Although SAS/ACCESS Interface to Greenplum defaults to three threads when you use autopartitioning, do not specify a maximum number of threads for the threaded Read in [DBSLICEPARM= LIBNAME option on page 226](#).

---

## Using DBSLICE=

You might achieve the best possible performance when using threaded Reads by specifying the [DBSLICE= data set option](#) for Greenplum in your SAS operation. This is especially true if your Greenplum data is evenly distributed across multiple partitions in a Greenplum database system.

When you create a Greenplum table using the Greenplum database partition model, you can specify the partitioning key that you want to use by appending the `PARTITION BY<column-name>` clause to your CREATE TABLE statement. Here is how you can accomplish this by using the [DB\\_CREATE\\_TABLE\\_OPTS=LIBNAME](#) option within the SAS environment.

```
/* Points to a triple-node server. */
libname mylib greenplm server=gplum03 user=myuser pw=mypwd
db=greenplum;
DBCREATE_TABLE_OPTS='PARTITION BY (EMPNUM)';

proc datasets library=mylib;
```

```

      delete MYEMPS1;run;

      data mylib.myemps(drop=morf whatstate
      DBTYPE=(HIREDATE="date" SALARY="numeric(8,2)"
      NUMCLASS="smallint" GENDER="char(1)" ISTENURE="numeric(1)"
      STATE="char(2)"
      EMPNUM="int NOT NULL Primary Key"));
      format HIREDATE mmddyy10.;
      do EMPNUM=1 to 100;
      morf=mod(EMPNUM,2)+1;
      if(morf eq 1) then
      GENDER='F';
      else
      GENDER='M';
      SALARY=(ranuni(0)*5000);
      HIREDATE=int(ranuni(13131)*3650);
      whatstate=int(EMPNUM/5);
      if(whatstate eq 1) then
      STATE='FL';
      if(whatstate eq 2) then
      STATE='GA';
      if(whatstate eq 3) then
      STATE='SC';
      if(whatstate eq 4) then
      STATE='VA';
      else
      state='NC';
      ISTENURE=mod(EMPNUM,2);
      NUMCLASS=int(EMPNUM/5)+2;
      output;
      end;
      run;

```

After the MYEMPS table is created on this three-node database, a third of the rows reside on each of the three nodes.

Using DBSLICE= works well when the table that you want to read is not stored in multiple partitions. It gives you flexibility in column selection. For example, if you know that the STATE column in your employee table contains only a few distinct values, you can modify your DBSLICE= option accordingly.

```

      data work.locemp;
      set mylib.MYEMPS (DBSLICE= ("STATE='GA'"
      "STATE='SC'" "STATE='VA'" "STATE='NC'"));
      where EMPNUM<=30 and ISTENURE=0 and SALARY<=35000 and NUMCLASS>2;
      run;

```

---

## Temporary Table Support for Greenplum

SAS/ACCESS Interface to Greenplum supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

---

# Passing SAS Functions to Greenplum

SAS/ACCESS Interface to Greenplum passes the following SAS functions to Greenplum for processing. Where the Greenplum function name differs from the SAS function name, the Greenplum name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                            |                    |
|----------------------------|--------------------|
| ** (POWER(base, exponent)) | LOWCASE (LOWER)    |
| ABS                        | MAX                |
| ARCOS (ACOS)               | MIN                |
| ARSIN (ASIN)               | MINUTE (EXTRACT)   |
| ATAN                       | MONTH (EXTRACT)    |
| ATAN2                      | QTR (EXTRACT)      |
| AVG                        | REPEAT             |
| BYTE (CHR)                 | SECOND (EXTRACT)   |
| CEIL                       | SIGN               |
| COMPRESS (TRANSLATE)       | SIN                |
| COS                        | SQRT               |
| COT                        | STRIP (BTRIM)      |
| COUNT                      | SUBSTR (SUBSTRING) |
| DAY (EXTRACT)              | SUM                |
| EXP                        | TAN                |
| FLOOR                      | TRANWRD (REPLACE)  |
| HOUR (EXTRACT)             | TRIMN (RTRIM)      |
| INDEX (STRPOS)             | UPCASE (UPPER)     |
| LENGTH                     | WEEKDAY (EXTRACT)  |
| LOG (LN)                   | YEAR (EXTRACT)     |
| LOG10 (LOG)                |                    |

`SQL_FUNCTIONS=ALL` allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when `SQL_FUNCTIONS=ALL` can the SAS/ACCESS engine also pass these SAS SQL functions to Greenplum. Due to incompatibility in date and time functions between Greenplum and SAS, Greenplum might not process them correctly. Check your results to determine whether these functions are working as expected.

|                    |                     |
|--------------------|---------------------|
| DATE (NOW)         | TIME (current_time) |
| DATEPART (CONVERT) | TIMEPART (TIME)     |
| DATETIME (NOW)     | TODAY (NOW)         |

---

## Passing Joins to Greenplum

For a multiple libref join to pass to Greenplum, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)
- host (HOST=)
- server (SERVER=)
- database (DATABASE=)
- port (PORT=)
- data source (DSN=, if specified)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see [“Passing Joins to the DBMS” on page 61](#).

---

## Sorting Data That Contains NULL Values

Some DBMSs, including Greenplum, place NULL values at the end of a sorted list. SAS normally places NULL values at the beginning of a sorted list. In SAS, to use BY-group processing, SAS expects that values for a BY group are already sorted before it performs BY-group processing. If SAS determines that values are not sorted, then BY-group processing stops. Therefore, if NULL values are not at the beginning of a sorted list, SAS determines that the values are not sorted correctly.

If you use PROC SQL with an ORDER BY statement to sort your data, then the SQL code is generated so that NULL values are placed at the beginning of the sorted list. In this way, SAS can perform any BY-group processing without stopping.

For more information, see [“Potential Result Set Differences When Processing Null Data” on page 43](#) and [“Sorting DBMS Data” on page 49](#).

---

# Bulk Loading for Greenplum

---

## Overview

Bulk loading provides high-performance access to external data sources. Multiple Greenplum instances read data in parallel, which enhances performance.

Bulk loading is the fastest way to insert large numbers of rows into Greenplum tables. You can also use bulk loading to execute high-performance SQL queries against external data sources, without first loading those data sources into a Greenplum database. These fast SQL queries let you optimize extraction, transformation, and loading tasks that are common in data warehousing.

Two types of external data sources, external tables and web tables, have different access methods. External tables contain static data that can be scanned multiple times. The data does not change during queries. Web tables provide access to dynamic data sources as if those sources were regular database tables. Web tables cannot be scanned multiple times. The data can change during the course of a query.

You must specify **BULKLOAD=YES** to use the bulk-load facility.

The following sections show you how to access external tables and web tables using the bulk-load facility.

---

## Using Protocols to Access External Tables

Use these protocols to access (static) external tables.

### gpfldist://

To use the gpfldist:// protocol, install and configure the gpfldist (Greenplum file distribution) program on the host that stores the external tables, see “[Configuring the File Server](#)”. The gpfldist utility serves external tables in parallel to the primary Greenplum database segments. The gpfldist:// protocol is advantageous because it ensures that all Greenplum database segments are used during the loading of external tables.

To specify files to gpfldist, use the **BL\_DATAFILE=** data set option. Specify file paths that are relative to the directory from which gpfldist is serving files (the directory where you executed gpfldist).

The gpfldist utility is part of the Greenplum Clients package for the platform where SAS is running. You can also download it from [VMware Tanzu Greenplum](#).

### file://

To use the file:// protocol, external tables must reside on a segment host in a location that Greenplum superusers (gpadmin) can access. The segment host name must match the host name, as specified in the gp\_configuration system

catalog table. In other words, the external tables that you want to load must reside on a host that is part of the set of servers that comprise the database configuration. The file:// protocol is advantageous because it does not require configuration.

## Configuring the File Server

Follow these steps to configure the gpfdist file server.

- 1 Download and install gpfdist from the Greenplum Clients package at [VMware Tanzu Greenplum](#).
- 2 Specify and load a new environment variable called GPLOAD\_HOME.
- 3 Set the value of the variable to the directory that contains the external tables that you want to load.

The directory path must be relative to the directory in which you execute gpfdist, and it must exist before gpfdist tries to access it.

- For Windows, open **My Computer**, select the **Advanced** tab, and click the **Environment Variables** button.
- For UNIX, enter this command or add it to your profile:

```
export GPLOAD_HOME=directory
```

- 4 Start gpfdist as shown in these examples.

- For Windows:

```
C:> gpfdist -d %GPLOAD_HOME% -p 8080 -l %GPLOAD_HOME%\gpfdist.log
```

- For UNIX:

```
$ gpfdist -d $GPLOAD_HOME -p 8080 -l $GPLOAD_HOME/gpfdist.log &
```

You can run multiple instances of gpfdist on the same host as long each instance has a unique port and directory.

If you do not specify GPLOAD\_HOME, the value of the BL\_DATAFILE= data set option specifies the directory that contains the external tables to be loaded. If BL\_DATAFILE is not specified, the current directory is assumed to contain the external tables.

## Stopping gpfdist

In Windows, to stop an instance of gpfdist, use the Task Manager or close the Command Window that you used to start that instance of gpfdist.

Follow these steps In UNIX to stop an instance of gpfdist.

- 1 Find the process ID:

```
$ ps ax | grep gpfdist (Linux)
$ ps -ef | grep gpfdist (Solaris)
```

- 
- 2 Kill the process. Here is an example:

```
$ kill 3456
```

---

## Troubleshooting gpfdist

Run this command to test connectivity between an instance of gpfdist and a Greenplum database segment.

```
$ wget http://gpfdist_hostname:port/file-name
```

---

## Using the file:// Protocol

You can use the file:// protocol to identify external files for bulk loading with no additional configuration required. However, using the GPOLOAD\_HOME environment variable is highly recommended. If you do not specify GPOLOAD\_HOME, the BL\_DATAFILE data set option specifies the source directory. The default source directory is the current directory if you do not specify BL\_DATAFILE=. The Greenplum server must have access to the source directory.

---

## Accessing Dynamic Data in Web Tables

Use these data set options to access web tables:

- [BL\\_EXECUTE\\_CMD=](#)
- [BL\\_LOCATION=](#)

---

## Data Set Options for Bulk Loading

Here are the Greenplum bulk-load data set options. For detailed information about these options, see [Data Set Options for Relational Databases on page 370](#).

- [BL\\_DATAFILE=](#)
- [BL\\_CLIENT\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DELIMITER=](#)
- [BL\\_ENCODING=](#)
- [BL\\_ESCAPE=](#)
- [BL\\_EXCEPTION=](#)
- [BL\\_EXECUTE\\_CMD=](#)

- BL\_EXECUTE\_LOCATION=
- BL\_EXTERNAL\_WEB=
- BL\_FORCE\_NOT\_NULL=
- BL\_FORMAT=
- BL\_HEADER=
- BL\_HOST=
- BL\_NULL=
- BL\_PORT=
- BL\_PROTOCOL=
- BL\_QUOTE=
- BL\_REJECT\_LIMIT=
- BL\_REJECT\_TYPE=
- BL\_USE\_PIPE=
- BULKLOAD=

## Bulk Loading Examples

This first example shows how you can use a SAS data set, SASFLT.FLT98, to create and load a Greenplum table, FLIGHTS98.

```

libname sasflt 'SAS-data-library';
libname mydblib greenplm server=mysrv1_users
      db=users user=myusr1 password=mypwd1;

proc sql;
  create table net_air.flights98
    (BULKLOAD=YES
     BL_DATAFILE='c:\temp\greenplum\data.dat'
     BL_USE_PIPE=NO
     BL_DELETE_DATAFILE=yes
     BL_HOST='192.168.x.x'
     BL_PORT=8081)
    as select * from sasflt.flt98;
quit;

```

This next example shows how you can append the SAS data set, SASFLT.FLT98, to the existing Greenplum table ALLFLIGHTS. The BL\_USE\_PIPE=NO option forces SAS/ACCESS Interface to Greenplum to write data to a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the load has completed.

```

proc append base=new_air.flights98
  (BULKLOAD=YES
   BL_DATAFILE='c:\temp\greenplum\data.dat'
   BL_USE_PIPE=NO
   BL_DELETE_DATAFILE=no
   BL_HOST='192.168.x.x'

```

```
BL_PORT=8081)
data=sasflt.flt98;
run;
```

# Locking in the Greenplum Interface

These LIBNAME and data set options let you control how the Greenplum interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134. For additional information, see your Greenplum documentation.

`READ_LOCK_TYPE= ROW (default) | TABLE`

`UPDATE_LOCK_TYPE= ROW (default) | TABLE`

`READ_ISOLATION_LEVEL= RC | RR | RU | S | V`

The Greenplum database manager supports the RC, RR, RU, S, and V isolation levels that are defined in the following table. The default value is RC.

Regardless of the isolation level, the database manager places exclusive locks on every row that is inserted, updated, or deleted. All isolation levels therefore ensure that only this application process can change any given row during a unit of work. No other application process can change any rows until the unit of work is complete.

*Table 21.3 Isolation Levels for Greenplum*

| Isolation Level       | Definition                                                                                                                                                                                                                                               |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RC (read uncommitted) | Allows dirty read, nonrepeatable read, and phantom Read operations.                                                                                                                                                                                      |
| RR (repeatable read)  | Does not allow dirty read, nonrepeatable read, or phantom Read operations.                                                                                                                                                                               |
| RU (read uncommitted) | Allows dirty read, nonrepeatable read, and phantom Read operations.                                                                                                                                                                                      |
| S (serializable)      | Does not allow dirty read, nonrepeatable read, or phantom Read operations.                                                                                                                                                                               |
| V (versioning)        | Does not allow dirty read, nonrepeatable read, or phantom Read operations. These transactions are serializable, but higher concurrency is possible with this level than with the serializable isolation level. Typically, a nonlocking protocol is used. |

Here is how the terms in the table are defined.

*Dirty reads*

A transaction that has very minimal isolation from concurrent transactions. In fact, this type of transaction can see changes that concurrent transactions make even before those transactions commit their changes.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

*Nonrepeatable reads*

A transaction where the system reads a row once and is unable to read the row again later in the same transaction. This might occur if a concurrent transaction changed or even deleted the row. Therefore, the Read operation is not (necessarily) repeatable.

For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

*Phantom reads*

A transaction where a set of rows that is read once might become a different set of rows if the transaction attempts to read the rows again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy a condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist (a “phantom”).

**UPDATE\_ISOLATION\_LEVEL= RC | RR | S | V**

The Greenplum database manager supports the RC, RR, S, and V isolation levels that are defined in the preceding table. Uncommitted reads are not allowed with this option. The default value is RC.

## Naming Conventions for Greenplum

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21..](#)

Most SAS names can be up to 32 characters long. The Greenplum interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name results in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The **PRESERVE\_TAB\_NAMES=** and **PRESERVE\_COL\_NAMES=** options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Greenplum is not case sensitive, so all names default to lowercase.

Greenplum objects include tables, views, and columns. They follow these naming conventions.

- A name can contain up to 128 characters.
- The first character in a name can be a letter, @, \_, or #.
- A name cannot be a Greenplum reserved word, such as WHERE or VIEW.
- A name must be unique within each type of each object.

---

# Data Types for Greenplum

---

## Overview

Every column in a table has a name and a data type. The data type tells Greenplum how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Greenplum data types, null and default values, and data conversions.

SAS/ACCESS Interface to Greenplum does not directly support any data types that are not listed below. Any columns using these types are read into SAS as character strings.

For more information about Greenplum data types and to determine which data types are available for your version of Greenplum, see your Greenplum documentation.

---

## Supported Greenplum Data Types

Here are the data types that SAS/ACCESS Interface to Greenplum supports:

- Character data:

CHAR(*n*)  
VARCHAR(*n*)  
TEXT

- Numeric data:

|                  |                         |
|------------------|-------------------------|
| BIGINT           | REAL                    |
| SMALLINT         | FLOAT                   |
| INTEGER          | DECIMAL   DEC   NUMERIC |
| DOUBLE PRECISION |                         |

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational

purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Date, time, and timestamp data:

DATE

TIME

TIMESTAMP

**Note:** Be aware that columns of these data types can contain data values that are out of range for SAS.

- Binary data: BYTEA

## Greenplum Null Values

Greenplum has a special value called NULL. A Greenplum NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Greenplum NULL value, it interprets it as a SAS missing value. When loading SAS tables from Greenplum sources, SAS/ACCESS stores Greenplum NULL values as SAS missing values.

In Greenplum tables, NULL values are valid in all columns by default. There are two methods to specify a column in a Greenplum table so that it requires data:

- Using SQL, you specify a column as NOT NULL. This tells SQL to allow only a row to be added to a table if a value exists for the field. Rows that contain NULL values in that column are not added to the table.
- Another approach is to assert NOT NULL DEFAULT.

When creating Greenplum tables with SAS/ACCESS, you can use the DBNULL= data set option to specify the treatment of NULL values. For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

Once you know whether a Greenplum column enables NULLs or the host system provides a default value for a column that is specified as NOT NULL WITH DEFAULT, you can write selection criteria and enter values to update a table. Unless a column is defined as NOT NULL or NOT NULL DEFAULT, it enables NULL values.

To control how the DBMS handles SAS missing character values, use the NULLCHAR= and [NULLCHARVAL=](#) data set options.

## LIBNAME Statement Data Conversions

The following table shows the default formats that SAS/ACCESS Interface to Greenplum assigns to SAS variables when using the [LIBNAME statement](#) to read from a Greenplum table.

These default formats are based on Greenplum column attributes.

**Table 21.4** LIBNAME Statement: Default SAS Formats for Greenplum Data Types

| Greenplum Data Type           | SAS Data Type | Default SAS Format |
|-------------------------------|---------------|--------------------|
| CHAR( $n$ ) <sup>1</sup>      | character     | \$w.               |
| VARCHAR( $n$ ) <sup>1</sup>   | character     | \$w.               |
| BYTEA                         | character     | \$w. <sup>3</sup>  |
| INTEGER                       | numeric       | 11.                |
| SMALLINT                      | numeric       | 6.                 |
| BIGINT                        | numeric       | 20.                |
| DECIMAL( $p,s$ ) <sup>2</sup> | numeric       | w.d                |
| NUMERIC( $p,s$ ) <sup>2</sup> | numeric       | w.d                |
| REAL                          | numeric       | none               |
| TIME                          | numeric       | TIME8.             |
| DATE                          | numeric       | DATE9.             |
| TIMESTAMP                     | numeric       | DATETIME25.6       |

<sup>1</sup>  $n$  in Greenplum data types is equivalent to  $w$  in SAS formats.

<sup>2</sup>  $p$  and  $s$  in Greenplum numeric data types are equivalent to  $w$  and  $d$  in SAS formats.

<sup>3</sup> Because the Greenplum ODBC driver does the conversion, this field is displayed as if the \$HEXw. format were applied.

The next table shows the default Greenplum data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 21.5** LIBNAME Statement: Default Greenplum Data Types for SAS Variable Formats

| SAS Variable Format | Greenplum Data Type           |
|---------------------|-------------------------------|
| w.d                 | DECIMAL( $p,s$ ) <sup>2</sup> |
| \$w.                | VARCHAR( $n$ ) <sup>1</sup>   |
| datetime formats    | TIMESTAMP                     |
| date formats        | DATE                          |

| SAS Variable Format | Greenplum Data Type |
|---------------------|---------------------|
| time formats        | TIME                |

1 *n* in Greenplum data types is equivalent to *w* in SAS formats.  
2 *p* and *s* in Greenplum numeric data types are equivalent to *w* and *d* in SAS formats.

---

## Sample Programs for Greenplum

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to Hadoop

---

|                                                                     |            |
|---------------------------------------------------------------------|------------|
| <b>System Requirements for SAS/ACCESS Interface to Hadoop .....</b> | <b>896</b> |
| <b>Introduction to SAS/ACCESS Interface to Hadoop .....</b>         | <b>896</b> |
| Hadoop Concepts .....                                               | 896        |
| Storing SAS Data on Your Hadoop Cluster .....                       | 897        |
| Hadoop Configuration .....                                          | 897        |
| National Language Support Limitations .....                         | 898        |
| <b>LIBNAME Statement for the Hadoop Engine .....</b>                | <b>898</b> |
| Overview .....                                                      | 898        |
| JDBC Read Security .....                                            | 899        |
| HDFS Write Security .....                                           | 899        |
| HDFS Permission Requirements for Optimized Reads .....              | 899        |
| Arguments .....                                                     | 900        |
| Hadoop LIBNAME Statement Examples .....                             | 904        |
| <b>Data Set Options for Hadoop .....</b>                            | <b>906</b> |
| <b>SQL Pass-Through Facility Specifics for Hadoop .....</b>         | <b>908</b> |
| Key Information .....                                               | 908        |
| CONNECT Statement Examples .....                                    | 908        |
| <b>Temporary Table Support for Hadoop .....</b>                     | <b>909</b> |
| <b>Passing SAS Functions to Hadoop .....</b>                        | <b>909</b> |
| <b>Passing Joins to Hadoop .....</b>                                | <b>910</b> |
| <b>Bulk Loading for Hadoop .....</b>                                | <b>911</b> |
| Loading .....                                                       | 911        |
| Examples .....                                                      | 912        |
| <b>Naming Conventions for SAS and Hive .....</b>                    | <b>912</b> |
| <b>Data Types for Hadoop .....</b>                                  | <b>913</b> |
| Supported Hive Data Types .....                                     | 913        |
| LIBNAME Statement Data Conversions .....                            | 914        |
| Issues When Converting Data from Hive to SAS .....                  | 915        |
| Leverage Table Properties for Existing Hive Tables .....            | 916        |

|                                                                                      |            |
|--------------------------------------------------------------------------------------|------------|
| Address Issues When Converting Data from Hive to SAS with Table Properties . . . . . | 918        |
| Alternatives to Table Properties for Issues with Data Conversion                     |            |
| from Hive to SAS . . . . .                                                           | 919        |
| Address Issues When Converting Data from Hive to SAS for Pass-Through SQL . . . . .  | 920        |
| Hadoop Null Values . . . . .                                                         | 921        |
| <b>Sample Programs for Hadoop . . . . .</b>                                          | <b>922</b> |
| Code Snippets . . . . .                                                              | 922        |
| Use DBSASTYPE= to Load Hadoop Data into SAS . . . . .                                | 922        |
| Create a Partitioned Table with a File Type of SEQUENCEFILE . . . . .                | 923        |

---

# System Requirements for SAS/ACCESS Interface to Hadoop

You can find information about system requirements for SAS/ACCESS Interface to Hadoop in the following locations:

- [System Requirements for SAS/ACCESS Interface to Hadoop with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

See also: "Configuring SAS/ACCESS Interface to Hadoop" in [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#)

---

# Introduction to SAS/ACCESS Interface to Hadoop

---

## Hadoop Concepts

Hadoop is a general-purpose data storage and computing platform that includes SQL-like products, such as Apache Hive. SAS/ACCESS Interface to Hadoop lets you work with your data by using HiveQL, a query language that is based on SQL. HiveQL has its own extensions and limitations.

With SAS/ACCESS Interface to Hadoop, you can read and write data to and from Hadoop as if it were any other relational data source to which SAS can connect. This interface provides fast, efficient access to data stored in Hadoop through HiveQL. You can access Hive tables as if they were native SAS data sets and then analyze them using SAS.

SAS/ACCESS Interface to Hadoop also reads data directly from the Hadoop Distributed File System (HDFS) when possible to improve performance. This differs from the traditional SAS/ACCESS engine behavior, which exclusively uses database SQL to read and write data. For more information, see the [READ\\_METHOD=LIBNAME](#) option.

When using HiveQL, the Hadoop engine requires access to the HiveServer2 service that runs on the Hadoop cluster, often using port 10000. For HDFS operations, such as writing data to Hive tables, the Hadoop engine requires access to the HDFS service that runs on the Hadoop cluster, often using port 8020. If the Hadoop engine cannot access the HDFS service, its full functionality is not available and the READ\_METHOD= option is constrained to READ\_METHOD=JDBC.

Beginning in [SAS Viya 3.2](#), SAS/ACCESS Interface to Hadoop includes SAS Data Connector to Hadoop. If you have the appropriate license, you might also have access to the SAS Data Connect Accelerator to Hadoop. The data connector or data connect accelerator enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“Hadoop Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

For available SAS/ACCESS features, see [Hadoop supported features](#) on page 103. For more information about Hadoop, see your Hadoop documentation.

## Storing SAS Data on Your Hadoop Cluster

SAS/ACCESS Interface to Hadoop has another mode of operation that does not use HiveQL. When you specify the [HDFS\\_METADIR=](#) connection option, SAS data sets are persisted on HDFS in a format that can be read directly by SAS. This is a useful way to store large amounts of SAS data on a low-cost Hadoop cluster.

Metadata about the SAS data set is persisted as a file with the SASHMDM file type. SAS/ACCESS creates SASHMDM metadata when it writes output from SAS. As an alternative, the HDMD procedure can create these metadata files.

For more information about using the HDMD procedure to make your HDFS files available to SAS, see the HDMD procedure in [Base SAS Procedures Guide](#).

## Hadoop Configuration

Your Hadoop administrator configures the Hadoop cluster that you use. Your administrator has specified the defaults for system parameters such as block size and replication factor that affect the Read and Write performance of your system. Replication factors greater than 1 help prevent data loss yet slow the writing of data. Consult with your administrator about how your particular cluster was configured.

To connect to a Hadoop server, you must complete these configuration steps.

- SAS/ACCESS Interface to Hadoop uses JDBC to submit Hive SQL requests to the Hadoop cluster. You can use the Apache Hive open source JDBC driver or a JDBC driver that is recommended by your Hadoop vendor. Drivers other than the Apache Hive open source driver must be configured for use.

- You must use a supported Hadoop distribution and configure a required set of Hadoop JAR files. The JAR files must be located in one location and must be available to the SAS client machine. The SAS environment variable `SAS_HADOOP_JAR_PATH` must be specified and point to the folders that contain all Hadoop JAR files.
- Hadoop cluster configuration files include `core-site.xml`, `hdfs-site.xml`, `hive-site.xml`, `mapred-site.xml`, and, if applicable, `yarn-site.xml`. You must copy Hadoop configuration files from the Hadoop cluster to a physical location that the SAS client machine can access. You must also define and set the SAS environment variable `SAS_HADOOP_CONFIG_PATH` to the location of the Hadoop configuration files. If you are using MapR, no `hdfs-site.xml` file is required in the directory.
- To connect to the Hadoop server through WebHDFS, the SAS environment variable `SAS_HADOOP_RESTFUL` must be specified and set to the value 1. In addition, the Hadoop `hdfs-site.xml` configuration file must include the properties for the WebHDFS location.

For more information, see the *SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS*.

## National Language Support Limitations

Keep these limitations in mind when working with Hadoop and Hive.

- Column names must use only WLATIN1 characters
- Table comments should also contain only WLATIN1 characters when the `DBCREATE_TABLE_OPTS=` data set option creates them using `DBCREATE_TABLE_OPTS="COMMENT 'my table comment'"`. Although this option accepts any NLS character, the NLS portion of the comment is not displayed properly later.

## LIBNAME Statement for the Hadoop Engine

### Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Hadoop supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

**Note:** SAS 9.4M8 expands support for JDBC drivers other than the Apache Hive open source driver. For more information, see "Configuring SAS/ACCESS Interface to Hadoop" in *SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS*.

Connections using third-party drivers are supported through the URI= LIBNAME option. The URI= option is the recommended way to connect to a Hadoop data source in SAS 9.4M8.

Here is the LIBNAME statement syntax for accessing Hadoop.

**LIBNAME** *libref* **hadoop** <*connection-options*> <*LIBNAME-options*>

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## JDBC Read Security

SAS/ACCESS can access Hadoop data through a JDBC connection to a HiveServer2 service. Depending on what release of Hive you have, Hive might not implement Read security. If Hive does not implement Read security, a successful connection from SAS allows Read access to all data that is accessible to the Hive service. SAS/ACCESS can connect to a Hive or HiveServer2 service that is unsecured, user name and password secured, or secured by Kerberos.

---

## HDFS Write Security

SAS/ACCESS creates and appends to Hive tables using the HDFS service. SAS/ACCESS can connect to a Hive or HiveServer2 service that is unsecured, user name and password secured, or secured by Kerberos. Your HDFS connection needs Write access to the HDFS /tmp directory. After data is written to /tmp, a Hive LOAD command is issued on your JDBC connection to associate the data with a Hive table. Therefore, the JDBC Hive session also needs Write access to /tmp.

**Note:** If HDFS /tmp has enabled the sticky bit, the LOAD command can fail. To resolve this, either disable the /tmp sticky bit or use the HDFS\_TEMPDIR option to specify an alternative HDFS directory for SAS/ACCESS to write data to.

---

---

## HDFS Permission Requirements for Optimized Reads

To optimize big data Reads, SAS/ACCESS creates a temporary table in HDFS /tmp. This requires that the SAS JDBC connection have Write access to /tmp. The temporary table is read using HDFS, so the SAS HDFS connection needs Read access to the temporary table that is written to /tmp. Alternatively, use the HDFS\_TEMPDIR option to specify an HDFS directory to use instead of /tmp.

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *hadoop*

specifies the SAS/ACCESS engine name for the Hadoop interface.

### *connection-options*

provides connection information and controls how SAS manages timing and concurrence of the connection to Hadoop. All connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

### `CLASSPATH="JAR-path"`

specifies the path to the JDBC JAR files. The value that you specify for CLASSPATH= overrides the value that is set for the SAS\_ACCESS\_CLASSPATH environment variable.

Default: SAS\_ACCESS\_CLASSPATH environment variable value

Requirement: You must set a value for the CLASSPATH= connection option or the SAS\_ACCESS\_CLASSPATH environment variable.

---

**Note:** Support for this option was added in SAS 9.4M8.

---

### `DRIVERCLASS="value"`

specifies the JDBC driver to use for your database connection.

Alias: DRIVER=, CLASS=, HIVECLASS=

Examples:

- Cloudera Hive JDBC Driver:

```
driverclass="com.cloudera.hive.jdbc.HS2Driver"
```

- Apache Hive JDBC Driver:

```
driverclass="org.apache.hive.jdbc.HiveDriver"
```

### `HDFS_DATADIR='path'`

when not in Hive mode, specifies the path to the Hadoop directory where SAS reads and writes data (for example, '/sas/hpa'). For details, see "Accessing Data Independently from Hive" in *Base SAS Procedures Guide* in *Base SAS Procedures Guide*.

Alias: HDFS\_PERMDIR=

---

**Note:** Use this option only when you are not using Hive or HiveServer2.

---

### `HDFS_METADIR='path'`

specifies the path to an HDFS directory that contains XML-based table definitions, called SASHMD descriptors. Through these descriptors, SAS then accesses the data using HDFS instead of Hive.

Use this option only when you are not using Hive or HiveServer2.

**HDFS\_TEMPDIR='path'**

specifies the path to the HDFS directory where SAS reads and writes temporary data.

Default: `HDFS_TEMPDIR= '/tmp'`

Use this option only when you are not using Hive or HiveServer2.

**PASSWORD=<'>Hadoop-password<'>**

specifies the Hadoop password that is associated with your user ID. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. If you do not want to enter your Hadoop password in uncoded text on this statement, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

Alias: `PASS=`, `PWD=`, `PW=`

**PORt=port**

specifies the port number to use to connect to the specified Hive service.

Alias: `SERVICE=`

Default: 10000

**SCHEMA=Hive-schema**

specifies the Hive schema.

Alias: `DATABASE=`, `DB=`

Default: default

**SERVER=<'>Hadoop-server-name<'>**

specifies the Hadoop server name that runs the Hive service. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: `HOST=`

**URI='jdbc:driver-name://driver-connection-options'**

specifies the full Hive JDBC connection string. The user's input is used exactly as it was specified to connect to Hive, and SAS does not modify the connection string. See the documentation for your JDBC driver for information about how to construct the connection string. See “[Hadoop LIBNAME Statement Examples](#)” on page 904 for examples of URL values.

Alias: `URL=`

Default: none

Restriction: You cannot use `URL=` in a LIBNAME statement when `HDFS_METADIR=` is present.

Requirement: Knox requires this option.

---

**Note:** The `URI=` option is required when using a JDBC driver other than the Apache Hive open source JDBC driver.

---

**USER=<'>Hadoop-user-name<'>**

specifies the user name for Read (JDBC) and Write (HDFS) operations. Do not use the `USER=` argument if your Hadoop cluster is secured by Kerberos.

Alias: `UID=`

*LIBNAME-options*

specifies how SAS processes DBMS objects. The following table describes the LIBNAME options for SAS/ACCESS Interface to Hadoop, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 22.1** SAS/ACCESS LIBNAME Options for Hadoop

| Option                   | Default Value | Valid in CONNECT |
|--------------------------|---------------|------------------|
| ACCESS=                  | none          |                  |
| ANALYZE=                 | NO            |                  |
| BL_FORMAT=               | TEXT          |                  |
| BL_PORT=                 | 8020          |                  |
| BULKLOAD=                | YES           |                  |
| CONNECTION=              | SHAREDREAD    | •                |
| DBCONINIT=               | none          | •                |
| DBCONTERM=               | none          | •                |
| DBCREATE_TABLE_EXTERNAL= | NO            |                  |
| DBCREATE_TABLE_LOCATION= | none          |                  |
| DBCREATE_TABLE_OPTS=     | none          |                  |
| DBGEN_NAME=              | DBMS          |                  |
| DBLIBINIT=               | none          |                  |
| DBLIBTERM=               | none          |                  |
| DBMAX_TEXT=              | 1024          | •                |
| DBMSTEMP=                | NO            |                  |
| DBSASLABEL=              | COMPAT        |                  |
| DEFER=                   | NO            | •                |
| DIRECT_SQL=              | YES           |                  |
| IGNORE_BASELINE=         | NO            | •                |

| Option              | Default Value                                | Valid in CONNECT |
|---------------------|----------------------------------------------|------------------|
| LOGIN_TIMEOUT=      | 30                                           | •                |
| MULTI_DATASRC_OPT=  | NONE                                         |                  |
| POST_STMT_OPTS=     | none                                         |                  |
| PRESERVE_COL_NAMES= | NO                                           |                  |
| PRESERVE_TAB_NAMES= | NO                                           |                  |
| PROPERTIES=         | none                                         | •                |
| QUERY_TIMEOUT=      | 0                                            | •                |
| READ_METHOD=        | none                                         | •                |
| READBUFF=           | automatically calculated based on row length | •                |
| SCHEMA=             | Hive schema default                          |                  |
| SCRATCH_DB=         | none                                         | •                |
| SPOOL=              | YES                                          |                  |
| SQL_FUNCTIONS=      | none                                         | •                |
| SQL_FUNCTIONS_COPY= | none                                         |                  |
| SUB_CHAR=           | none                                         |                  |
| TEMP_CTAS           | YES                                          |                  |
| TRANSCODE_FAIL=     | ERROR                                        | •                |

## Hadoop LIBNAME Statement Examples

### Connecting to Hive Using a Third-Party Driver

This example connects to Hive using the Cloudera JDBC driver for Apache Hive. HDFS access is achieved through Java based APIs. Important things to know about the connection are as follows:

- The user must download the Cloudera JDBC driver for Apache Hive from the Cloudera website and copy it to the location `/data-drivers/jdbc`. The LIBNAME option `DRIVERCLASS='com.cloudera.hive.jdbc.HS2Driver'` can be specified in the LIBNAME statement to force the loading of the driver.

---

**Note:** Starting in SAS 9.4M8, resources that are copied to the `/data-drivers/jdbc` location are automatically added to the SAS/ACCESS product's internal class path. For earlier releases of SAS, use the `SAS_ACCESS_CLASSPATH` environment variable or `CLASSPATH= LIBNAME` statement option to specify the location of the JDBC driver. (Use the `SET=` system option to define the SAS environment variable.)

---

- The administrator has previously run the Hadoop tracer tool and copied the Hadoop JAR and configuration files to the locations that were specified by the `SAS_HADOOP_JAR_PATH=` and `SAS_HADOOP_CONFIG_PATH=` environment variables.
- SAS/ACCESS Interface to Hadoop bulk loads data to Hadoop using HDFS. `BULKLOAD=YES` is the default; therefore, there is no need to specify this LIBNAME option.
- The JDBC URL that is specified in the `URL= LIBNAME` option specifies options for the Cloudera JDBC driver for Hive. For more information, see the [Cloudera](#) documentation.
- When using the Cloudera JDBC driver for Hive, you must specify `useNativeQuery=1` in the JDBC URL, as SAS/ACCESS to Hadoop already generates native HiveQL syntax.
- You might also want to specify `DefaultStringColumnLength=` to limit the length of the STRING columns read into SAS.

```
option set=SAS_HADOOP_JAR_PATH="/mysrv1_jars";
option set=SAS_HADOOP_CONFIG_PATH="/mysrv1_cfg";

libname x hadoop user='user-name' pwd='password'
driver_class='com.cloudera.hive.jdbc.HS2Driver'
url='jdbc:hive2://<server>:<port>/
<database>;useNativeQuery=1,defaultStringColumnLength=1024'
;
```

---

## Connecting to Hive with RESTful HDFS Access (Knox)

In this example, HDFS is accessed through WebHDFS or another REST-based API. This type of connection has the advantage of not requiring any JAR files to be pulled from the Hadoop cluster, although Hadoop configuration files are still required. Important things to know about the connection are as follows:

- The administrator has previously run the Hadoop tracer tool and copied the Hadoop configuration files to the locations specified by the SAS\_HADOOP\_CONFIG\_PATH= environment variable.
- When you use Knox, you must also define the KNOX\_GATEWAY\_URL= environment variable and you must set the SAS\_HADOOP\_RESTFUL environment variable to 1.
- As in the previous example, the JDBC URL specifies driver options for the Cloudera JDBC driver for Hive.

Specify useNativeQuery=1 in the JDBC URL, because SAS generates native Hive SQL.

```

options set=SAS_HADOOP_RESTFUL=1;
options set=SAS_HADOOP_CONFIG_PATH="/mysrv1_cfg";
options set=KNOX_GATEWAY_URL="cloudera-URL";
options set=SAS_HADOOP_JAR_PATH="/mysrv1_jars";

libname x hadoop user=user-name pw='password' bulkload=yes
driverclass=com.cloudera.hive.jdbc.HS2Driver
uri="jdbc:hive2://cloudera-server-id:port/
database;transportMode=http;ssl=1;httpPath=myHttpPath;useNativeQuery=1;
"
;
```

---

## Connecting to Hive with Kerberos Authentication

This example is similar to the first example except Kerberos is used for authentication. Before Kerberos can be used, an administrator must first set up Kerberos for JDBC connections using “kinit”. For instructions, see information about SAS/ACCESS Interface to Hadoop in [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#). In this example, the Kerberos service principal name is `hive/node1.example.com@EXAMPLE.COM`, the host name for the data source is `node1.example.com`, and the server is listening on port 1000 for JDBC connections.

```

options set=SAS_HADOOP_JAR_PATH="/mysrv1_jars";
options set=SAS_HADOOP_CONFIG_PATH="/mysrv1_cfg";

%kerberos_kinit_domain(database);

libname x hadoop
```

```

driverclass=com.cloudera.hive.jdbc.HS2Driver
url='jdbc:hive2://node1.example.com:10000/
default;AuthMech=1;KrbRealm=example.com;
KrbHostFQDN=hs2node1.example.com;KrbServiceName=hive'
;
```

## Connecting to Hive without HDFS Access

In this example, HDFS is not available. All data access is done through JDBC. Important things to know about the connection are as follows:

- Environment variables SAS\_HADOOP\_CONFIG\_PATH= and SAS\_HADOOP\_JAR\_PATH= are not required.
- BULKLOAD=NO is specified so that SAS does not write the bulk load staging file to HDFS.
- The READ\_METHOD=JDBC LIBNAME option is specified in the LIBNAME statement so that SAS does not attempt to materialize any result set to HDFS.

```

libname x hadoop user='user-name' pw='password'
bulkload=no read_method=jdbc
uri='jdbc:hive2://<server>:<port>/
<database>;useNativeQuery=1;defaultStringColumnLength=1024'
;
```

## Data Set Options for Hadoop

All SAS/ACCESS data set options in this table are supported for Hadoop. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 22.2** SAS/ACCESS Data Set Options for Hadoop

| Option                     | Default Value        |
|----------------------------|----------------------|
| ANALYZE= data set option   | LIBNAME option value |
| BL_FORMAT= data set option | TEXT                 |
| BULKLOAD=                  | YES                  |
| COLUMN_DELIMITER=          | \001 (Ctrl-A)        |
| DBCONDITION=               | none                 |
| DBCREATE_TABLE_EXTERNAL=   | NO                   |

| Option                   | Default Value                                         |
|--------------------------|-------------------------------------------------------|
| DBCREATE_TABLE_LOCATION= | depends on your Hadoop cluster configuration          |
| DBCREATE_TABLE_OPTS=     | LIBNAME option value                                  |
| DBFORCE=                 | NO                                                    |
| DBGEN_NAME=              | DBMS                                                  |
| DBLARGETABLE=            | none                                                  |
| DBMAX_TEXT=              | 1024                                                  |
| DBNULL=                  | none                                                  |
| DBSASLABEL=              | COMPAT                                                |
| DBSASTYPE=               | see <a href="#">Data Types for Hadoop on page 913</a> |
| DBTYPE=                  | see <a href="#">Data Types for Hadoop on page 913</a> |
| POST_STMT_OPTS=          | none                                                  |
| POST_TABLE_OPTS=         | none                                                  |
| PRE_STMT_OPTS=           | none                                                  |
| PRE_TABLE_OPTS=          | none                                                  |
| QUERY_TIMEOUT=           | LIBNAME option value                                  |
| READ_METHOD=             | none                                                  |
| READBUFF=                | LIBNAME option value                                  |
| SCHEMA=                  | LIBNAME option value                                  |
| SCRATCH_DB=              | none                                                  |
| TRANSCODE_FAIL=          | ERROR                                                 |

---

# SQL Pass-Through Facility Specifics for Hadoop

---

## Key Information

For general information about this feature, see “[Overview of SQL Procedure Interactions with SAS/ACCESS](#)” on page 689.

Here are the SQL pass-through facility specifics for the Hadoop interface.

- The *dbms-name* is HADOOP.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Hadoop. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default HADOOP alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).

---

## CONNECT Statement Examples

---

### Connect Using an Already Assigned Libref

When a Hadoop libref is already assigned, you can use PROC SQL’s CONNECT USING syntax to avoid re-typing the connection options.

```
/* libref x is already assigned */
proc sql;
  connect using x;
  /* get the HiveQL version */
  select * from connection to x (select version () );
  /* create a table using HiveQL, not SAS SQL */
  execute (create table if not exists test1 (i integer) ) by x;
  disconnect from x;
quit;
```

---

## Connect Using Connection Parameters

If a Hadoop libref is not available, you can connect to Hadoop using PROC SQL's CONNECT TO syntax. You must provide the same connection parameters that are required in the LIBNAME statement in the CONNECT TO statement.

This example uses the CData JDBC driver for Apache Hive.

```
proc sql;
  connect to hadoop as x (
    driverClass="cdata.jdbc.hive.ApacheHiveDriver"
    url='jdbc:apachehive:Server=cdp71p1hive;QueryPassthrough=True';
    Database=mydatabase;DefaultColumnSize=1024'
    user=myuser1
    password=mypwd1
  );
  /* get the HiveQL version */
  select * from connection to x (select version () );
  /* create a table using HiveQL, not SAS SQL */
  execute (create table if not exists test1 (i integer) ) by x;
  disconnect from x;
quit;
```

---

## Temporary Table Support for Hadoop

SAS/ACCESS Interface to Hadoop supports temporary tables. For more information, see [“Temporary Table Support for SAS/ACCESS” on page 51](#).

---

## Passing SAS Functions to Hadoop

SAS/ACCESS Interface to Hadoop passes the following SAS functions to Hadoop for processing. Where the Hadoop function name differs from the SAS function name, the Hadoop name appears in parentheses. For more information, see [“Passing Functions to the DBMS Using PROC SQL” on page 58](#).

|              |                 |
|--------------|-----------------|
| ** (POWER)   | LOG (LN)        |
| ABS          | LOG2            |
| ARCOS (ACOS) | LOG10           |
| ARSIN (ASIN) | LOWCASE (LOWER) |
| ATAN         | MAX             |
| AVG          | MIN             |
| CEIL         | MOD             |

|                            |                      |
|----------------------------|----------------------|
| COMPRESS (REGEXP_REPLACE)* | MONTH                |
| COS                        | SCAN (REGEX_SPLIT)** |
| CAT (CONCAT)               | SIGN                 |
| COUNT                      | SIN                  |
| DAY                        | SQRT                 |
| DTEXTDAY (DAY)             | STD (STDDEV_SAMP)    |
| DTEXTMONTH (MONTH)         | STRIP (TRIM)         |
| DTEXTWEEKDAY               | SUBSTR               |
| DTEXTYEAR (YEAR)           | SUM                  |
| EXP                        | TAN                  |
| FLOOR                      | TRIMN (RTRIM)        |
| INDEX (LOCATE)             | UPCASE (UPPER)       |
| LEFT (LTRIM)               | VAR (VAR_SAMP)       |
| LENGTH (LENGTH)            | YEAR                 |
| LENGTHN (LENGTH)           |                      |

\* Only when you specify two arguments—for example, COMPRESS(string,' ').

\*\* Only when you specify two arguments where the second argument is nonzero—for example, SCAN (*expression*, 1).

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding Hadoop functions that are passed down to Hadoop. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to Hadoop. Due to incompatibility in date and time functions between Hadoop and SAS, Hadoop might not process them correctly. Check your results to determine whether these functions are working as expected.

|                                             |                                    |
|---------------------------------------------|------------------------------------|
| COALESCE                                    | QUARTER                            |
| DATE (TO_DATE(UNIX_TIMESTAMP))              | REPEAT                             |
| DATEPART<br>(TO_DATE(UNIX_TIMESTAMP))       | ROUND                              |
| DATETIME<br>(FROM_UNIXTIME(UNIX_TIMESTAMP)) | SECOND                             |
| HOUR                                        | SOUNDEX                            |
| MINUTE                                      | TODAY<br>(TO_DATE(UNIX_TIMESTAMP)) |
| QTR                                         | TRANSTRN<br>(REGEXP_REPLACE)       |

## Passing Joins to Hadoop

For a multiple libref join to pass to Hadoop, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)

- server (SERVER=)
- port (PORT=)
- schema (SCHEMA=)

For more information, see “[Passing Joins to the DBMS](#)” on page 61.

# Bulk Loading for Hadoop

## Loading

When BULKLOAD=YES is specified (the default value), SAS creates a “staging file” that contains the data to be loaded into Hive, and then loads the data into Hive by issuing the Hive LOAD DATA query.

The exact sequence of commands varies, depending on the file formats of the target table and the staging file.

If the target table does not exist, the table's file format can be specified using the [DBCREATE\\_TABLE\\_OPTS= LIBNAME option](#) or [data set option](#). When this option is not used, Hive creates the target table using its default file format. The Hive default file format is specified by the Hive configuration option `hive.default.fileformat` property.

The staging file's format can be different from the target table's file format. Starting in SAS 9.4M8, you can specify the file format of the staging file with the `BL_FORMAT=` option. For example, you can create the Hive table with a file format of Parquet by specifying `DBCREATE_TABLE_OPTS="STORED AS PARQUET"` and use `BL_FORMAT="ORC"` to use ORC when writing the staging file. For more information, see [BL\\_FORMAT= LIBNAME option](#) or [data set option](#). Hive transforms the data from the staging file into the target table's file format as part of the load operation.

New tables are created in the Hive warehouse. To create tables outside of the Hive warehouse, use the [DBCREATE\\_TABLE\\_EXTERNAL=](#) and [DBCREATE\\_TABLE\\_LOCATION=](#) options. These options are available as LIBNAME and data set options.

Here is the general sequence of operations:

- 1 If the target table does not exist, SAS submits a CREATE TABLE query to Hive to create the target table. The CREATE TABLE query contains the column definitions, any Hive Table Properties, and any SAS LIBNAME options or data set options that are specified.
- 2 SAS creates the staging file that contains the data to be loaded to Hive.
  - When `BL_FORMAT=TEXT`, SAS writes the data to the HDFS temporary directory. The staging file is created as a UTF-8 delimited text file, using CTRL-A ('\001') as a field delimiter and newline ('\n') as a record separator.
  - When `BL_FORMAT=` is not TEXT, the staging file is created locally on the machine where SAS is running and then copied to HDFS.

- 3 SAS issues a CREATE TABLE query to create a temporary table in Hive and then issues a LOAD DATA query to load the staging file into the temporary table. This temporary table is referred to as the “staging table.”
- 4 SAS issues an INSERT INTO *target-table* SELECT \* FROM query, which copies the data from the staging table to the target table.
- 5 SAS issues a DROP TABLE query to drop the temporary staging table.

In some limited cases, SAS can perform the LOAD DATA operation directly into the target table.

When PROC APPEND is used to append to the Hive table, the Hadoop interface places data in a new HDFS file. The interface then issues either the LOAD DATA pattern or the LOAD DATA plus INSERT INTO pattern described earlier.

As of SAS 9.4M6, SAS/ACCESS Interface to Hadoop also supports BULKLOAD=NO. When BULKLOAD=NO, SAS creates an SQL INSERT INTO ... VALUES query to send to Hive. Although this approach is slower than using BULKLOAD=YES, BULKLOAD=NO is useful when HDFS Write access is not available.

## Examples

This example creates and loads the FLIGHTS98 HiveServer2 table from the SASFLT.FLT98 SAS data set.

```
libname sasflt 'SAS-library';
libname hdp_air hadoop user=myusr1 pwd=mypwd1 connection-options;

proc sql;
create table hdp_air.flights98
      as select * from sasflt.flt98;
quit;
```

This example creates and loads the ALLFLIGHTS HiveServer2 table in SEQUENCEFILE format from the SASFLT.ALLFLIGHTS SAS data set.

```
data hdp_air.allflights (dbcreate_table_opts='stored as sequencefile');
  set sasflt.allflights;
run;
```

In this example, the SASFLT.FLT98 SAS data set is appended to the ALLFLIGHTS HiveServer2 table.

```
proc append base=hdp_air.allflights data=sasflt.flt98;run;
```

## Naming Conventions for SAS and Hive

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names”](#).

Current versions of Hive are case insensitive for comparisons, but case is preserved for display purposes. Users can set the PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options (shortcut PRESERVE\_NAMES=) to preserve the case of identifiers. Doing this usually is not required unless the case must be preserved for display purposes.

Hive currently does not permit a period (.) or colon (:) within a column name, and a table name cannot begin with the underscore (\_) character.

SAS and Hadoop objects include tables, views, columns, and indexes. Here is how SAS handles Hive names:

- A SAS name must be from 1 to 32 characters long. When Hive column names and table names are 32 characters or less, SAS handles them seamlessly. When SAS reads Hive column names that are longer than 32 characters, a generated SAS variable name is truncated to 32 characters. Hive table names should be 32 characters or less because SAS cannot truncate a table reference. If you already have a table name that is greater than 32 characters, create a Hive table view or use the explicit SQL feature of PROC SQL to access the table.
- If truncating would result in identical names, SAS generates a unique name.
- For National Language Support, because of Hive limitations, column names must use only WLATIN1 characters. Table comments should also contain only WLATIN1 characters when the DBCREATE\_TABLE\_OPTS= data set option creates them using DBCREATE\_TABLE\_OPTS="COMMENT 'my table comment'". Although this option accepts any NLS character, the NLS portion of the comment is not displayed properly later.

## Data Types for Hadoop

### Supported Hive Data Types

Here are the Hive data types that the Hadoop engine supports.

- Numeric data

|         |          |
|---------|----------|
| BIGINT  | FLOAT    |
| BOOLEAN | INT      |
| DECIMAL | SMALLINT |
| DOUBLE  | TINYINT  |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the

data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Character data

|          |             |
|----------|-------------|
| BINARY   | STRING      |
| CHAR $n$ | VARCHAR $n$ |

- Date and time data

|          |           |
|----------|-----------|
| DATE     | TIMESTAMP |
| INTERVAL |           |

- Complex data

|       |        |
|-------|--------|
| ARRAY | STRUCT |
| MAP   |        |

---

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Hadoop assigns to SAS variables when using the “[LIBNAME Statement for the Hadoop Engine](#)”.. These default formats are based on Hive column attributes.

**Table 22.3 LIBNAME Statement: Default SAS Formats for Hive Data Types**

| Hive Data Type                | SAS Data Type | Default SAS Format |
|-------------------------------|---------------|--------------------|
| ARRAY                         | character     | none               |
| BINARY                        | character     | \$HEX32767         |
| BOOLEAN                       | numeric       | w. (1.)            |
| CHAR( $n$ ) <sup>1</sup>      | character     | \$w.               |
| DATE                          | numeric       | DATE9.             |
| DECIMAL( $p,s$ ) <sup>2</sup> | numeric       | w.d                |
| DOUBLE                        | numeric       | w.                 |
| FLOAT                         | numeric       | w.                 |
| INT                           | numeric       | w. (11.)           |
| INTERVAL                      | numeric       | DATETIME25.6       |
| MAP                           | character     | none               |

| Hive Data Type                   | SAS Data Type | Default SAS Format |
|----------------------------------|---------------|--------------------|
| SMALLINT                         | numeric       | w. (6.)            |
| STRING <sup>3</sup>              | character     | \$32767.           |
| STRUCT                           | character     | none               |
| TIMESTAMP                        | numeric       | DATETIME25.6       |
| TINYINT                          | numeric       | w. (4.)            |
| VARCHAR( <i>n</i> ) <sup>1</sup> | character     | \$w.               |

<sup>1</sup> *n* in Hive data types is equivalent to *w.* in SAS formats.

<sup>2</sup> *p,s* in Hive data types is equivalent to *w.d* in SAS formats

<sup>3</sup> The STRING data type might be mapped to the VARCHAR data type by the JDBC client driver. For more information, check your JDBC vendor's documentation.

This table shows the default Hive data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 22.4 LIBNAME Statement: Default Hive Data Types for SAS Variable Formats**

| SAS Variable Format       | Hive Data Type                 |
|---------------------------|--------------------------------|
| <i>w.d</i>                | DOUBLE                         |
| <i>w.</i>                 | INT, SMALLINT, TINYINT, BIGINT |
| \$ <i>w.</i>              | VARCHAR                        |
| datetime formats          | TIMESTAMP                      |
| date formats              | DATE                           |
| time formats <sup>1</sup> | VARCHAR                        |

<sup>1</sup> A column created in Hive using a TIME format is created as a VARCHAR column with a SASFMT format.

## Issues When Converting Data from Hive to SAS

Below are some potential conversion issues.

- Hive STRING columns that contain ANSI date, time, or timestamp values do not automatically convert respectively to SAS DATE, TIME, or DATETIME types.
- STRING: Depending on the length of Hadoop STRING data, the SAS character \$32767. format might be unnecessarily large for short STRING columns or can

truncate Hadoop STRING columns that contain more than 32767 characters. To specify a limit for an individual STRING column, issue a Hive ALTER TABLE statement limiting the column length. Here is an example: ALTER TABLE weblogs SET TBLPROPERTIES ('SASFMT:webdata'='CHAR(1000)'). To specify a general limit for multiple STRING columns, use the DBMAX\_TEXT= option.

- BIGINT: Converting Hadoop BIGINT to a SAS numeric can result in loss of precision because the internal SAS eight-byte, floating-point format accurately preserves only 15 digits of precision. A BIGINT preserves up to 19.

Work-arounds are based on how you access data.

- explicitly using pass-through SQL. See “Address Issues When Converting Data from Hive to SAS for Pass-Through SQL” on page 920.
- using the LIBNAME statement. See “Address Issues When Converting Data from Hive to SAS with Table Properties” on page 918.

## Leverage Table Properties for Existing Hive Tables

SAS/ACCESS generates SAS table properties only when creating a new Hive table. Many or perhaps all of your Hive tables are created by other means. For example, your Hadoop administrator might create Hive table definitions by submitting DDL scripts to the Hive CLI. SAS and SAS users can benefit by adding SAS table properties to existing Hive table definitions. In this example, a Hive table has already been specified.

```
CREATE EXTERNAL TABLE weblogs (extract_date STRING,
    extract_type INT, webdata STRING) ROW FORMAT DELIMITED FIELDS
    TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/user/hadoop/web_data'
```

Based on this table definition, here is how SAS interprets the columns.

```
libname hdp hadoop server=mysrv1 user=myusr1 pwd=mypwd1;
data sheetmetal_sales; set hdp.weblogs(obs=1);
put extract_date= extract_type=;
put webdata=;
run;
```

```
extract_date=2012-02-21 extract_type=1
webdata=http://www.sas.com/industry/oilgas
NOTE: There were 1 observations read from the data set HDP.WEBLOGS.
```

```
proc contents data=hdp.weblogs; run;
```

| Alphabetic List of Variables and Attributes |              |      |       |          |          |              |
|---------------------------------------------|--------------|------|-------|----------|----------|--------------|
| #                                           | Variable     | Type | Len   | Format   | Informat | Label        |
| 1                                           | extract_date | Char | 32767 | \$32767. | \$32767. | extract_date |
| 2                                           | extract_type | Num  | 8     | 11.      | 11.      | extract_type |
| 3                                           | webdata      | Char | 32767 | \$32767. | \$32767. | webdata      |

Notice that Hive describes the `extract_date` column to SAS as a 32767 length STRING. It also describes the `webdata` column as a 32767 length STRING. So SAS/ACCESS enters both of these columns as character data and uses \$32767. to

format them. The result is an overly wide SAS data set with an observation (row) width of 64 kilobytes that also does not format `extract_date` to a SAS DATE.

SAS issues a warning message for this situation, which includes the maximum column length that was in the result set. In the example, the maximum length read for the `extract_date` STRING column is 10 bytes. The maximum length read for the `webdata` STRING column was 320 bytes.

```
WARNING: SAS/ACCESS assigned these columns a length of 32767. If
resulting SAS character variables remain this length, SAS performance
is impacted. See SAS/ACCESS documentation for details. Columns
followed by the maximum length observed were: extract_date:10,
webdata:320
```

The example below assumes that the length of the `webdata` STRING in Hive never exceeds 1000 characters. A Hadoop user ID with the appropriate authority can issue Hive ALTER TABLE statements to add SAS table properties to the Hive table definition.

```
ALTER TABLE weblogs SET TBLPROPERTIES ('SASFMT:extract_date='DATE(9.0)')
ALTER TABLE weblogs SET TBLPROPERTIES ('SASFMT:webdata='CHAR(1000)')
```

SAS/ACCESS recognizes the added properties and here is the result.

```
libname hdp hadoop server=mysrv1 user=myusr1 pwd=mypwd1;
data sheetmetal_sales; set hdp.weblogs(obs=1);
put extract_date= extract_type=;
put webdata=;
run;
```

```
extract_date=21FEB2012 extract_type=1
webdata=http://www.sas.com/industry/oilgas
NOTE: There were 1 observations read from the data set HDP.WEBLOGS.
```

```
proc contents data=hdp.weblogs; run;
```

| Alphabetic List of Variables and Attributes |              |      |      |        |          |              |
|---------------------------------------------|--------------|------|------|--------|----------|--------------|
| #                                           | Variable     | Type | Len  | Format | Informat | Label        |
| 1                                           | extract_date | Num  | 8    | DATE9. | DATE9.   | extract_date |
| 2                                           | extract_type | Num  | 8    | 11.    | 11.      | extract_type |
| 3                                           | webdata      | Char | 1000 | \$1000 | \$1000.  | webdata      |

The resulting SAS data set that is created from the Hive table has a much smaller observation width, which helps SAS save disk space and reduce CPU consumption. It also automatically converts and formats `extract_date` to SAS standard DATE9. format.

Adding SAS properties to existing Hive tables does not affect table use by software that is not SAS. You can also issue ALTER TABLE and other DDL statements using SAS/ACCESS explicit SQL.. (See “[SQL Pass-Through Facility Specifics for Hadoop](#)” on page 908.) Issuing such DDL as an ALTER TABLE statement can be restricted to only the Hadoop administrator.

## Address Issues When Converting Data from Hive to SAS with Table Properties

Some issues currently exist when reading Hadoop data into SAS. (See “[Issues When Converting Data from Hive to SAS](#)” on page 915.) For example, Hive STRING columns default to \$32767. SAS character format without a specified SASFMT table property or a SAS override option such as DBSASTYPE=.

Here is how you can address specific conversion issues.

### STRING issues

To automatically convert Hive STRING columns that contain ANSI date, timestamp, or time values to suitable SAS formats, you can use the following HiveQL ALTER TABLE statements. In the statements are these sample Hive columns: `d` contains ANSI date, `ts` contains ANSI timestamp, and `t` contains ANSI time.

---

**Note:** Keep in mind that you cannot run ALTER TABLE commands in SAS because they are not SAS syntax. You must execute these commands within Hive.

---

```
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:d'='DATE(9.0)')
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:ts'='DATETIME(25.6)')
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:t'='TIME(15.6)')
```

Instead, you could use these statements to create SAS character variables of optimal length that contain the identical ANSI representation as those that are stored in Hive:

```
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:d'='CHAR(9)')
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:ts'='CHAR(25)')
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:t'='CHAR(15)')
```

You can use the following statement for other Hive STRING columns where the maximum length is less than 32767. Here, the `string_col` column has a maximum length of 100.

```
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:string_col'='CHAR(100)')
```

However, if you anticipate that the Hive `string_col` column might grow to a maximum length of 200 in the future, you could instead use this statement to specify the table property.

```
ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:string_col'='CHAR(200)')
```

Hive STRING columns longer than 32767 characters are truncated when they are read into SAS. Here is how the warning for this data loss is flagged in the SAS log:

```
WARNING: Column 'string_col' was truncated 1 times.
Observation (row) number 2 was the first observation truncated.
```

### BIGINT issues

Converting a Hadoop BIGINT column to a SAS numeric column can cause a loss of precision. A SAS numeric column can accurately preserve only 15 digits of precision. However, a BIGINT column can preserve up to 19 significant digits

of precision, plus one character for the possible sign (+/-). You can address this issue by applying a CHAR(20) table property format. SAS then automatically reads a Hive BIGINT column into a SAS character string with \$20. format. This format preserves all BIGINT digits in character format. Here is an example, using the `b gint` BIGINT column.

```
ALTER TABLE sample_table SET TBLPROPERTIES ('SASFMT:b gint='CHAR(20)')
```

Keep this important consideration in mind, however: For Hive tables that SAS/ACCESS creates, you might not need to issue ALTER TABLE statements.

---

**CAUTION**

**Do not create multiple table properties for a single Hive column.**

Unpredictable data conversion can result.

---

## Alternatives to Table Properties for Issues with Data Conversion from Hive to SAS

For various reasons, it might be impractical or undesirable to issue ALTER TABLE statements to create SAS table properties. In such cases, you can instead use these data set options.

**DBSASTYPE=**

Use DBSASTYPE= in your SAS code to cause data conversion from Hive to SAS that is identical to automatic conversion with table properties. The pairs below are SAS DATA steps with identical behavior. The first of each pair assumes a SASFMT table property, the second one assumes no table property, and DBSASTYPE= is added to achieve the same functionality. (For details, see the [DBSASTYPE= data set option](#).)

Here is the SAS LIBNAME statement for all of these SAS DATA steps.

---

**Note:** Remember that you cannot run ALTER TABLE commands in SAS because they are not SAS syntax. You must execute these commands within Hive.

---

```
libname hdp hadoop server=mysrv1 user=myusr1 pwd=mypwd1;

ALTER TABLE sample_table SET TBLPROPERTIES ('SASFMT:d='DATE(9.0)')
[---assumes table property 'SASFMT:d='DATE(9.0)' ---]
data work.local_sample; set hdp.sample_table( keep=d ); run;

[---assumes no table property for column 'd'---]
data work.local_sample;
set hdp.sample_table( keep=d dbsastype=(d='DATE(9.0)' ) ); run;

ALTER TABLE sample_table SET TBLPROPERTIES ('SASFMT:ts='DATETIME(25.6)')
[---assumes table property 'SASFMT:ts='DATETIME(25.6)' ---]
data work.local_sample; set hdp.sample_table( keep=ts ); run;

[---assumes no table property for column 'ts'---]
data work.local_sample;
```

```

set hdp.sample_table( keep=ts dbsastype=(ts='DATETIME(25.6)') ) ; run;

ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:t='TIME(15.6)')
[---assumes table property 'SASFMT:t='TIME(15.6)' ---]
data work.local_sample; set hdp.sample_table( keep=t ) ; run;

[---assumes no table property for column 't'---]
data work.local_sample;
set hdp.sample_table( keep=t dbsastype=(t='TIME(15.6)') ) ; run;

ALTER TABLE sample_table SET_TBLPROPERTIES ('SASFMT:string_col='CHAR(200)')
[---assumes table property 'SASFMT:string_col='CHAR(200)' ---]
data work.local_sample; set hdp.sample_table( keep=string_col ) ; run;

[---assumes no table property for column 'string_col'---]
data work.local_sample;
set hdp.sample_table( keep=string_col dbsastype=(string_col='CHAR(200)') ) ;
run;

```

**DBMAX\_TEXT=[n]**

You can use the DBMAX\_TEXT= option to limit the SAS length of all STRING columns read from Hive. For example, if you specify DBMAX\_TEXT=100, then all SAS character variables that are created from Hive STRING columns are limited to width \$100. Specifying the DBMAX\_TEXT= option likewise limits the length in SAS of Hive 12 and higher CHAR and VARCHAR columns.

## Address Issues When Converting Data from Hive to SAS for Pass-Through SQL

Neither table properties nor DBSASTYPE= address data conversion issues from Hive to SAS if you use pass-through SQL to read Hive data. For pass-through SQL, you might need to explicitly convert and format each Hive column as you want it to be represented in SAS. For example, suppose you use SAS to create a table with SAS table properties that are generated for all but the BIGINT column. Here is the table that SAS creates.

```

libname hdp hadoop server=mysrv1 user=myusr1 pwd=mypwd1;
data hdp.passthrough_ex( dbtype=(bgint="BIGINT") );
bgint='1234567890123456789';
format ts datetime25.6; ts=datetime();
format d date9.; d=today();
format t time10.; t=time();
format string_col $20.; string_col='hello';
run;

```

SAS issues this HiveQL when creating the table.

```

CREATE TABLE `PASSTHROUGH_EX`(`bgint` BIGINT,`ts` STRING,
`d` STRING,`t` STRING,`string_col` STRING) ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\001' STORED AS TEXTFILE TBLPROPERTIES
('SASFMT:ts='DATETIME(25.6)', 'SASFMT:d='DATE(9.0)',
'SASFMT:t='TIME(10.0)', 'SASFMT:string_col='CHAR(20))
```

Next, an ALTER TABLE statement is issued to add a table property for **BIGINT column bgint**.

```
ALTER TABLE passthrough_ex SET TBLPROPERTIES ('SASFMT:bgint='CHAR(20)')
```

A LIBNAME-based table that is read to SAS recognizes the table properties.

```
data work.local; set hdp.passthrough_ex; run;

data _null_; set work.local;
put bgint=; put ts=; put d=; put t=; put string_col=;
run;
```

```
bgint=1234567890123456789
ts=25FEB2012:02:00:55.141000
d=25FEB2012
t=2:00:55
string_col=hello
```

This pass-through SQL step converts and formats each column identically to the LIBNAME-based step that applied the table properties.

```
proc sql; connect to hadoop(server=mysrv1 user=myusr1 pwd=mypwd1);
create table work.local as select
bgint length 20 format $20. informat $20.,
input(ts, IS8601DT26.) as ts format datetime25.6 informat datetime25.6,
input(d, yymmdd10.) as d format date9. informat date9.,
input(t, IS8601TM15.) as t format time15.6 informat time15.6,
string_col length 20 format $20. informat $20.
from connection to hadoop( select cast(bgint as STRING)
as bgint,ts,d,t,string_col from passthrough_ex );
quit;
data _null_; set work.local;
put bgint=; put ts=; put d=; put t=; put string_col=;
run;
```

```
bgint=1234567890123456789
ts=25FEB2012:02:00:55.141000
d=25FEB2012
t=2:00:55.141000
string_col=hello
```

If SAS detects that a column length for a numeric variable is 32767 and could be less, it writes a message:

```
WARNING: These columns could have a length in SAS of 32767. If so,
SAS performance is impacted. See SAS/ACCESS documentation for
details. The columns read from Hive followed by the maximum length
observed were: bgint:20, ts:26, d:9, t:15, string_col:20.
```

## Hadoop Null Values

Hadoop has a special value called NULL. A Hadoop NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Hadoop NULL value, it interprets it as a SAS missing value. For more information

about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

# Sample Programs for Hadoop

## Code Snippets

The code snippets in this section resemble those for most other SAS/ACCESS interfaces.

This snippet shows a list of available Hive tables.

```
proc datasets lib=hdp; quit;
```

Here is the metadata for the mytab Hive table.

```
proc contents data=hdp.mytab; quit;
```

This snippet extracts mytab data into SAS.

```
data work.a;
set hdp.mytab;
run;
```

This extracts a subset of the mytab rows and columns into SAS. Subsetting the rows (with a WHERE statement, for example) can help avoid extracting too much data into SAS.

```
data work.a;
set hdp.mytab (keep=col1 col2);
where col2=10;
run;
```

## Use DBSASTYPE= to Load Hadoop Data into SAS

This example uses the [DBSASTYPE= data set option](#) to load Hadoop textual dates, timestamps, and times into the corresponding SAS DATE, DATETIME, and TIME formats. The first step reads in a SAS character string to display the data and make clear what occurs in successive steps.

```
data; set hdp.testHiveDate; put dt; run;
```

|                     |
|---------------------|
| 2011-10-17          |
| 2009-07-30 12:58:59 |
| 11:30:01            |

```
data; set hdp.testHiveDate(dbsastype=(dt='date')); put dt; run;
```

```
17OCT2011
30JUL2009
.
```

```
data; set hdp.testHiveDate(dbsastype=(dt='datetime')); put dt; run;
```

```
17OCT2011:00:00:00
30JUL2009:12:58:59
.
```

```
data; set hdp.testHiveDate(dbsastype=(dt='time')); put dt; run;
```

```
.
12:58:59
11:30:01
```

This code uses SAS SQL to access a Hadoop table.

```
proc sql;
create table work.a as select * from hdp.newtab;
quit;
```

SAS data is then loaded into Hadoop.

```
data hdp.newtab2;
set work.a;
run;
```

Use implicit pass-through SQL to extract only 10 rows from the Newtab table and load the work SAS data set with the results.

```
proc sql;
connect to hadoop server=hxpduped user=myusr1 password=mypwd1;
create table work.a as
select * from connection to hadoop (select * from newtab limit 10);
```

## Create a Partitioned Table with a File Type of SEQUENCEFILE

Use the DBCREATE\_TABLE\_OPTS value PARTITIONED BY (*column data-type*) STORED AS SEQUENCEFILE.

```
libname hdp HADOOP server=hxpduped user=myusr1 password=mypwd1;

data hdp.part_tab (DBCREATE_TABLE_OPTS="PARTITIONED BY (s2 int, s3
string)
STORED AS SEQUENCEFILE");
set work.part_tab;
run;
```

Sample programs for SAS/ACCESS Interface to Hadoop are available from <https://github.com/sassoftware/sas-access-samples>.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be

found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to HAWQ

---

|                                                             |     |
|-------------------------------------------------------------|-----|
| <i>System Requirements for SAS/ACCESS Interface to HAWQ</i> | 926 |
| <i>Introduction to SAS/ACCESS Interface to HAWQ</i>         | 926 |
| <b><i>LIBNAME Statement for the HAWQ Engine</i></b>         | 927 |
| Overview                                                    | 927 |
| Arguments                                                   | 927 |
| LIBNAME Statement Examples                                  | 931 |
| <b><i>Data Set Options for HAWQ</i></b>                     | 931 |
| <b><i>SQL Pass-Through Facility Specifics for HAWQ</i></b>  | 934 |
| Key Information                                             | 934 |
| CONNECT Statement Example                                   | 935 |
| Special Catalog Queries                                     | 935 |
| <b><i>Autopartitioning Scheme for HAWQ</i></b>              | 936 |
| Overview                                                    | 936 |
| Autopartitioning Restrictions                               | 936 |
| Nullable Columns                                            | 937 |
| Using WHERE Clauses                                         | 937 |
| Using DBSLICEPARM=                                          | 937 |
| Using DBSLICE=                                              | 937 |
| <b><i>Passing SAS Functions to HAWQ</i></b>                 | 938 |
| <b><i>Passing Joins to HAWQ</i></b>                         | 939 |
| <b><i>Sorting Data That Contains NULL Values</i></b>        | 940 |
| <b><i>Bulk Loading for HAWQ</i></b>                         | 940 |
| Overview                                                    | 940 |
| Using Protocols to Access External Tables                   | 941 |
| Configuring the File Server                                 | 941 |
| Stopping gpfdist                                            | 942 |
| Troubleshooting gpfdist                                     | 942 |
| Using the file:// Protocol                                  | 942 |
| Accessing Dynamic Data in Web Tables                        | 942 |
| Data Set Options for Bulk Loading                           | 943 |
| Bulk Loading Examples                                       | 943 |
| <b><i>Locking in the HAWQ Interface</i></b>                 | 944 |

|                                    |            |
|------------------------------------|------------|
| <i>Naming Conventions for HAWQ</i> | <b>946</b> |
| <i>Data Types for HAWQ</i>         | <b>946</b> |
| Overview                           | 946        |
| Supported HAWQ Data Types          | 947        |
| HAWQ Null Values                   | 947        |
| LIBNAME Statement Data Conversions | 948        |
| <i>Sample Programs for HAWQ</i>    | <b>949</b> |

---

# System Requirements for SAS/ACCESS Interface to HAWQ

You can find information about system requirements for SAS/ACCESS Interface to HAWQ in the following locations:

- [System Requirements for SAS/ACCESS Interface to HAWQ with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to HAWQ

For available SAS/ACCESS features, see [HAWQ supported features on page 104](#). For more information about HAWQ, see your HAWQ documentation.

---

**Note:** Starting with [SAS 9.4M8](#), SAS/ACCESS Interface to HAWQ is no longer available. If you have an existing installation of SAS/ACCESS Interface to HAWQ and plan to upgrade to SAS 9.4M8 or later, SAS recommends that you first unconfigure and uninstall SAS/ACCESS Interface to HAWQ. For more information, see [Unconfiguring and Uninstalling Retired Products](#). As an alternative to HAWQ, consider storing your data on Greenplum and accessing it with SAS/ACCESS Interface to Greenplum.

---

SAS/ACCESS Interface to HAWQ and SAS/ACCESS Interface to Greenplum both use the underlying Greenplum ODBC client. For this reason, you might see messages that refer to Greenplum in the SAS log.

---

# LIBNAME Statement for the HAWQ Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to HAWQ supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing HAWQ.

**LIBNAME** *libref hawq <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

*hawq*

specifies the SAS/ACCESS engine name for the HAWQ interface.

*connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the HAWQ database in two ways. Specify only one of these methods for each connection because they are mutually exclusive.

- SERVER=, DATABASE=, PORT=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**SERVER=<'>*server-name*<'>**

specifies the HAWQ server name or the IP address of the server host. If the server name contains spaces or nonalphanumeric characters or if it is an IP address, you must enclose it in quotation marks.

Alias: HOST=

**DATABASE=<'>*database-name*<'**

specifies the HAWQ database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORT=*port***

specifies the port number that is used to connect to the specified HAWQ database. If you do not specify a port, the default is 5432.

Aliases: SERVICE=, SERVICE\_NAME=

**USER=<'>*HAWQ-user-name*<'**

specifies the HAWQ user name (also called the user ID) that is used to connect to the database. If the user name contains spaces or nonalphanumeric characters, use quotation marks.

Alias: UID=

**PASSWORD=<'>*HAWQ-password*<'**

specifies the password that is associated with your HAWQ user ID. If the password contains spaces or nonalphabetic characters, you must enclose it in quotation marks. You can also specify PASSWORD= with the PWD=, PASS=, and PW= aliases.

**DSN=<'>*HAWQ-data-source*<'**

specifies the configured HAWQ ODBC data source to which you want to connect. It is recommended that you use this option only if you have configured HAWQ ODBC data sources on your client. This method requires additional setup—either through the ODBC Administrator control panel on Windows platforms, or through the odbc.ini file or a similarly named configuration file on UNIX platforms. It is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

Aliases: DS=, DATASRC=

***LIBNAME* -options**

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to HAWQ with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 23.1** SAS/ACCESS LIBNAME Options for HAWQ

| Option      | Default Value      | Valid in CONNECT |
|-------------|--------------------|------------------|
| ACCESS=     | none               |                  |
| AUTHDOMAIN= | none               |                  |
| AUTOCOMMIT= | operation-specific | •                |
| BULKLOAD=   | NO                 | •                |

| Option                    | Default Value                                      | Valid in CONNECT |
|---------------------------|----------------------------------------------------|------------------|
| CONNECTION=               | SHAREDREAD                                         | •                |
| CONNECTION_GROUP=         | none                                               | •                |
| CURSOR_TYPE=              | FORWARD_ONLY                                       | •                |
| DBCOMMIT=                 | 1000 (when inserting rows), 0 (when updating rows) |                  |
| DBCONINIT=                | none                                               | •                |
| DBCONTERM=                | none                                               | •                |
| DBCREATE_TABLE_OPTS=      | none                                               |                  |
| DBGEN_NAME=               | DBMS                                               | •                |
| DBINDEX=                  | NO                                                 |                  |
| DBLIBINIT=                | none                                               |                  |
| DBLIBTERM=                | none                                               |                  |
| DBMAX_TEXT=               | 1024                                               | •                |
| DBMSTEMP=                 | none                                               |                  |
| DBNULLKEYS=               | YES                                                |                  |
| DBPROMPT=                 | none                                               | •                |
| DBSASLLABEL=              | COMPAT                                             |                  |
| DBSLICEPARM=              | THREADED_APPS                                      |                  |
| DEFER=                    | none                                               | •                |
| DELETE_MULT_ROWS=         | NO                                                 |                  |
| DIRECT_EXE=               | none                                               |                  |
| DIRECT_SQL=               | YES                                                |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                 |                  |
| IN=                       | none                                               |                  |

| Option                | Default Value                                | Valid in CONNECT |
|-----------------------|----------------------------------------------|------------------|
| INSERT_SQL=           | YES                                          |                  |
| INSERTBUFF=           | automatically calculated based on row length |                  |
| KEYSET_SIZE=          | 0                                            | •                |
| MULTI_DATASRC_OPT=    | NONE                                         |                  |
| POST_STMT_OPTS=       | none                                         |                  |
| PRESERVE_COL_NAMES=   | YES<br>See “Naming Conventions for HAWQ”.    |                  |
| PRESERVE_TAB_NAMES=   | see “Naming Conventions for HAWQ”            |                  |
| QUERY_TIMEOUT=        | 0                                            | •                |
| QUOTE_CHAR=           | none                                         |                  |
| READ_ISOLATION_LEVEL= | RC                                           | •                |
| READ_LOCK_TYPE=       | ROW                                          | •                |
| READBUFF=             | automatically calculated based on row length | •                |
| REREAD_EXPOSURE=      | none                                         | •                |
| SCHEMA=               | none                                         |                  |
| SPOOL=                | YES                                          |                  |
| SQL_FUNCTIONS=        | none                                         |                  |
| SQL_FUNCTIONS_COPY=   | none                                         |                  |
| SQLGENERATION=        | none                                         |                  |
| STRINGDATES=          | NO                                           | •                |
| SUB_CHAR=             | none                                         |                  |

| Option                  | Default Value | Valid in CONNECT |
|-------------------------|---------------|------------------|
| TRACE=                  | none          | •                |
| TRACEFILE=              | none          | •                |
| UPDATE_ISOLATION_LEVEL= | RC            |                  |
| UPDATE_LOCK_TYPE=       | ROW           |                  |
| UPDATE_SQL=             | YES           | •                |
| UPDATE_MULT_ROWS=       | NO            |                  |
| UTILCONN_TRANSIENT=     | none          |                  |

---

## LIBNAME Statement Examples

In this example, SERVER=, DATABASE=, PORT=, USER=, and PASSWORD= are the connection options.

```
libname mydblib hawq server=hwq04 db=customers port=5432
      user=hwqusr1 password=hwqpwd1;

proc print data=mydblib.customers;
  where state='CA';
run;
```

In the next example, DSN=, USER=, and PASSWORD= are the connection options. The HAWQ data source is configured in the ODBC Administrator Control Panel on Windows platforms. It is also configured in the odbc.ini file or a similarly named configuration file on UNIX platforms.

```
libname mydblib hawq DSN=hwqSalesDiv user=hwqusr1 password=hwqpwd1;

proc print data=mydblib.customers;
  where state='CA';
run;
```

---

## Data Set Options for HAWQ

All SAS/ACCESS data set options in this table are supported for HAWQ. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 23.2** SAS/ACCESS Data Set Options for HAWQ

| Option               | Default Value                                    |
|----------------------|--------------------------------------------------|
| BL_DATAFILE_EXISTS=  | NO                                               |
| BL_DATAFILE=         | none                                             |
| BL_DELETE_DATAFILE=  | none                                             |
| BL_DELIMITER=        |                                                  |
| BL_ENCODING=         | DEFAULT                                          |
| BL_ESCAPE=           | \                                                |
| BL_EXCEPTION=        | none                                             |
| BL_EXECUTE_CMD=      | none                                             |
| BL_EXECUTE_LOCATION= | none                                             |
| BL_EXTERNAL_WEB=     | NO                                               |
| BL_FORCE_NOT_NULL=   | none                                             |
| BL_FORMAT=           | TEXT                                             |
| BL_HEADER=           | NO                                               |
| BL_HOST=             | 127.0.0.1                                        |
| BL_NULL=             | 'N' [TEXT mode], unquoted empty value [CSV mode] |
| BL_PORT=             | 8080                                             |
| BL_PROTOCOL=         | 'gpfdist'                                        |
| BL_QUOTE=            | " (double quotation mark)                        |
| BL_REJECT_LIMIT=     | none                                             |
| BL_REJECT_TYPE=      | ROWS                                             |
| BULKLOAD=            | none                                             |
| DBCOMMIT=            | LIBNAME option value                             |
| DBCONDITION=         | none                                             |

| Option                    | Default Value             |
|---------------------------|---------------------------|
| DBCREATE_TABLE_OPTS=      | LIBNAME option value      |
| DBFORCE=                  | none                      |
| DBGEN_NAME=               | DBMS                      |
| DBINDEX=                  | LIBNAME option value      |
| DBKEY=                    | none                      |
| DBLABEL=                  | none                      |
| DBLARGETABLE=             | none                      |
| DBMAX_TEXT=               | 1024                      |
| DBNULL=                   | none                      |
| DBNULLKEYS=               | LIBNAME option value      |
| DBPROMPT=                 | LIBNAME option value      |
| DBSASTYPE=                | see “Data Types for HAWQ” |
| DBSLICE=                  | none                      |
| DBSLICEPARM=              | THREADED_APPS             |
| DBTYPE=                   | see “Data Types for HAWQ” |
| DISTRIBUTED_BY=           | DISTRIBUTED_RANDOMLY      |
| ERRLIMIT=                 | 1                         |
| IGNORE_READ_ONLY_COLUMNS= | LIBNAME option value      |
| INSERTBUFF=               | LIBNAME option value      |
| KEYSET_SIZE=              | LIBNAME option value      |
| NULLCHAR=                 | SAS                       |
| NULLCHARVAL=              | a blank character         |
| POST_STMT_OPTS=           | none                      |
| POST_TABLE_OPTS=          | none                      |

| Option                  | Default Value        |
|-------------------------|----------------------|
| PRE_STMT_OPTS=          | none                 |
| PRE_TABLE_OPTS=         | none                 |
| PRESERVE_COL_NAMES=     | LIBNAME option value |
| QUERY_TIMEOUT=          | LIBNAME option value |
| READ_ISOLATION_LEVEL=   | LIBNAME option value |
| READ_LOCK_TYPE=         | LIBNAME option value |
| READBUFF=               | LIBNAME option value |
| SASDATEFMT=             | none                 |
| SCHEMA=                 | LIBNAME option value |
| SUB_CHAR=               | none                 |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |
| UPDATE_LOCK_TYPE=       | LIBNAME option value |

---

## SQL Pass-Through Facility Specifics for HAWQ

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the HAWQ interface.

- The *dbms-name* is `HAWQ`.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to HAWQ. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default `HAWQ` alias is used.

- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME connection options.

## CONNECT Statement Example

This example uses the DBCON alias to connect to the HAWQ04 HAWQ server database and execute a query. The connection alias is optional.

```
proc sql;
  connect to hawq as dbcon
    (server=HAWQ04 db=sample port=5432 user=hwqusr1 password=hwpwd1);
  select * from connection to dbcon
    (select * from customers where customer like '1%');
  quit;
```

## Special Catalog Queries

SAS/ACCESS Interface to HAWQ supports the following special queries. You can use the queries to call functions in ODBC-style function application programming interfaces (APIs). Here is the general format of the special queries:

HAWQ::SQLAPI '*parameter-1*', '*parameter-n*'

HAWQ::

is required to distinguish special queries from regular queries. HAWQ:: is not case sensitive.

SQLAPI

is the specific API that is being called. SQLAPI is not case sensitive.

'*parameter n*'

is a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, this call to SQLTables usually matches table names such as myatest and my\_test:

```
select * from connection to hawq
  (HAWQ::SQLTables "test", "", "my_test");
```

Use the escape character to search only for the my\_test table:

```
select * from connection to hawq
  (HAWQ::SQLTables "test", "", "my\_test");
```

SAS/ACCESS Interface to HAWQ supports these special queries.

HAWQ::SQLTables <'Catalog', 'Schema', 'Table-name', 'Type'>

returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.

HAWQ::SQLColumns <'Catalog', 'Schema', 'Table-name', 'Column-name'>  
 returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.

HAWQ::SQLColumns <'Catalog', 'Schema', 'Table-name', 'Column-name'>  
 returns a list of all columns that match the specified arguments. If you do not specify any argument, all accessible column names and information are returned.

HAWQ::SQLPrimaryKeys <'Catalog', 'Schema', 'Table-name' 'Type'>  
 returns a list of all columns that compose the primary key that matches the specified table. A primary key can be composed of one or more columns. If you do not specify any table name, this special query fails.

HAWQ::SQLStatistics <'Catalog', 'Schema', 'Table-name'>  
 returns a list of the statistics for the specified table name. You can specify the SQL\_INDEX\_ALL and SQL\_ENSURE options in the SQLStatistics API call. If you do not specify any table name argument, this special query fails.

HAWQ::SQLGetTypeInfo  
 returns information about the data types that the HAWQ database supports.

## Autopartitioning Scheme for HAWQ

### Overview

Autopartitioning for SAS/ACCESS Interface to HAWQ is a modulo (MOD) function method. For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

### Autopartitioning Restrictions

SAS/ACCESS Interface to HAWQ places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- INTEGER and SMALLINT columns are given preference.
- You can use other numeric columns for partitioning if the precision minus the scale of the column is greater than 0 but less than 10—namely,  $0 < (\text{precision} - \text{scale}) < 10$ .

---

## Nullable Columns

If you select a nullable column for autopartitioning, the `OR<column-name>IS NULL` SQL statement is appended at the end of the SQL code that is generated for the threaded Read. This ensures that any possible NULL values are returned in the result set.

---

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in a SAS WHERE clause. For example, this DATA step cannot use a threaded Read to retrieve the data because all numeric columns in the table are in the WHERE clause:

```
data work.locemp;
  set trlib.MYEMPS;
  where EMPNUM<=30 and ISTEMURE=0 and
    SALARY<=35000 and NUMCLASS>2;
run;
```

---

## Using DBSLICEPARM=

Although SAS/ACCESS Interface to HAWQ defaults to three threads when you use autopartitioning, do not specify a maximum number of threads for the threaded Read in [DBSLICEPARM= LIBNAME option on page 226](#).

---

## Using DBSLICE=

You might achieve the best possible performance when using threaded Reads by specifying the [DBSLICE= data set option](#) for HAWQ in your SAS operation. This is especially true if your HAWQ data is evenly distributed across multiple partitions in a HAWQ database system.

When you create a HAWQ table using the HAWQ database partition model, you can specify the partitioning key that you want to use by appending the `PARTITION BY<column-name>` clause to your CREATE TABLE statement. Here is how you can accomplish this by using the [DB\\_CREATE\\_TABLE\\_OPTS=LIBNAME](#) option within the SAS environment.

```
/* Points to a triple-node server. */
libname mylib hawq server=hawq03 user=myuser pw=mypwd db=HAWQ;
DBCREATE_TABLE_OPTS='PARTITION BY (EMPNUM)';

proc datasets library=mylib;
  delete MYEMPS1;run;
```

```

data mylib.myemps(drop=morf whatstate
DBTYPE=(HIREDATE="date" SALARY="numeric(8,2)"
NUMCLASS="smallint" GENDER="char(1)" ISTENURE="numeric(1)"
STATE="char(2)"
EMPNUM="int NOT NULL Primary Key"));
format HIREDATE mmddyy10.;
do EMPNUM=1 to 100;
morf=mod(EMPNUM,2)+1;
if(morf eq 1) then
GENDER='F';
else
GENDER='M';
SALARY=(ranuni(0)*5000);
HIREDATE=int(ranuni(13131)*3650);
whatstate=int(EMPNUM/5);
if(whatstate eq 1) then
STATE='FL';
if(whatstate eq 2) then
STATE='GA';
if(whatstate eq 3) then
STATE='SC';
if(whatstate eq 4) then
STATE='VA';
else
state='NC';
ISTENURE=mod(EMPNUM,2);
NUMCLASS=int(EMPNUM/5)+2;
output;
end;
run;

```

After the MYEMPS table is created on this three-node database, a third of the rows reside on each of the three nodes.

Using DBSLICE= works well when the table that you want to read is not stored in multiple partitions. It gives you flexibility in column selection. For example, if you know that the STATE column in your employee table contains only a few distinct values, you can modify your DBSLICE= option accordingly.

```

data work.locemp;
set mylib.MYEMPS (DBSLICE= ("STATE='GA'"
"STATE='SC'" "STATE='VA'" "STATE='NC'"));
where EMPNUM<=30 and ISTENURE=0 and SALARY<=35000 and NUMCLASS>2;
run;

```

## Passing SAS Functions to HAWQ

SAS/ACCESS Interface to HAWQ passes the following SAS functions to HAWQ for processing. Where the HAWQ function name differs from the SAS function name, the HAWQ name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                      |                    |
|----------------------|--------------------|
| ABS                  | MAX                |
| ARCOS (ACOS)         | MIN                |
| ARSIN (ASIN)         | MINUTE (DATEPART)  |
| ATAN                 | MONTH (DATEPART)   |
| ATAN2                | QTR (DATEPART)     |
| AVG                  | REPEAT             |
| BYTE (CHR)           | SECOND (DATEPART)  |
| CEIL                 | SIGN               |
| COMPRESS (TRANSLATE) | SIN                |
| COS                  | SQRT               |
| COUNT                | STRIP (BTRIM)      |
| DAY (DATEPART)       | SUBSTR (SUBSTRING) |
| EXP                  | SUM                |
| FLOOR                | TAN                |
| HOUR (DATEPART)      | TRANWRD (REPLACE)  |
| INDEX (STRPOS)       | TRIMN (RTRIM)      |
| LENGTH               | UPCASE (UPPER)     |
| LOG (LN)             | WEEKDAY (DATEPART) |
| LOG10 (LOG)          | YEAR (DATEPART)    |
| LOWCASE (LOWER)      |                    |

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to HAWQ. Due to incompatibility in date and time functions between HAWQ and SAS, HAWQ might not process them correctly. Check your results to determine whether these functions are working as expected.

|                    |                     |
|--------------------|---------------------|
| DATE (NOW)         | TIME (current_time) |
| DATEPART (CONVERT) | TIMEPART (TIME)     |
| DATETIME (NOW)     | TODAY (NOW)         |

## Passing Joins to HAWQ

For a multiple libref join to pass to HAWQ, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)
- host (HOST=)
- server (SERVER=)
- database (DATABASE=)
- port (PORT=)

- data source (DSN=, if specified)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Sorting Data That Contains NULL Values

Some DBMSs, including HAWQ, place NULL values at the end of a sorted list. SAS normally places NULL values at the beginning of a sorted list. In SAS, to use BY-group processing, SAS expects that values for a BY group are already sorted before it performs BY-group processing. If SAS determines that values are not sorted, then BY-group processing stops. Therefore, if NULL values are not at the beginning of a sorted list, SAS determines that the values are not sorted correctly.

If you use PROC SQL with an ORDER BY statement to sort your data, then the SQL code is generated so that NULL values are placed at the beginning of the sorted list. In this way, SAS can perform any BY-group processing without stopping.

For more information, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43 and “[Sorting DBMS Data](#)” on page 49.

---

## Bulk Loading for HAWQ

---

### Overview

Bulk loading provides high-performance access to external data sources. Multiple HAWQ instances read data in parallel, which enhances performance.

Bulk loading is the fastest way to insert large numbers of rows into HAWQ tables. You can also use bulk loading to execute high-performance SQL queries against external data sources, without first loading those data sources into a HAWQ database. These fast SQL queries let you optimize extraction, transformation, and loading tasks that are common in data warehousing.

Two types of external data sources, external tables and web tables, have different access methods. External tables contain static data that can be scanned multiple times. The data does not change during queries. Web tables provide access to dynamic data sources as if those sources were regular database tables. Web tables cannot be scanned multiple times. The data can change during the course of a query.

You must specify **BULKLOAD=YES** to use the bulk-load facility.

The following sections show you how to access external tables and web tables using the bulk-load facility.

---

## Using Protocols to Access External Tables

Use these protocols to access (static) external tables.

### gpfldist://

To use the gpfldist:// protocol, install and configure the gpfldist (HAWQ file distribution) program on the host that stores the external tables, see “[Configuring the File Server](#)”. The gpfldist utility serves external tables in parallel to the primary HAWQ database segments. The gpfldist:// protocol is advantageous because it ensures that all HAWQ database segments are used during the loading of external tables.

To specify files to gpfldist, use the BL\_DATAFILE= data set option. Specify file paths that are relative to the directory from which gpfldist is serving files (the directory where you executed gpfldist).

The gpfldist utility is part of the loader package for the platform where SAS is running. You can also download it from the HAWQ website: <http://pivotal.io/>.

### file://

To use the file:// protocol, external tables must reside on a segment host in a location that HAWQ superusers can access. The segment host name must match the host name, as specified in the gp\_configuration system catalog table. In other words, the external tables that you want to load must reside on a host that is part of the set of servers that comprise the database configuration. The file:// protocol is advantageous because it does not require configuration.

---

## Configuring the File Server

Follow these steps to configure the gpfldist file server.

- 1 Download and install gpfldist from <http://pivotal.io/>.
- 2 Specify and load a new environment variable called GPOLOAD\_HOME.
- 3 Set the value of the variable to the directory that contains the external tables that you want to load.

The directory path must be relative to the directory in which you execute gpfldist, and it must exist before gpfldist tries to access it.

- For Windows, open **My Computer**, select the **Advanced** tab, and click the **Environment Variables** button.
- For UNIX, enter this command or add it to your profile:  

```
export GPOLOAD_HOME=directory
```

- 4 Start gpfldist as shown in these examples.

- For Windows:

```
C:> gpfldist -d %GPOLOAD_HOME% -p 8080 -l %GPOLOAD_HOME%\gpfldist.log
```

- For UNIX:

```
$ gpfdist -d $GPOLOAD_HOME -p 8080 -l $GPOLOAD_HOME/gpfdist.log &
```

You can run multiple instances of gpfdist on the same host as long each instance has a unique port and directory.

If you do not specify GPOLOAD\_HOME, the value of the BL\_DATAFILE= data set option specifies the directory that contains the external tables to be loaded. If BL\_DATAFILE is not specified, the current directory is assumed to contain the external tables.

## Stopping gpfdist

In Windows, to stop an instance of gpfdist, use the Task Manager or close the Command Window that you used to start that instance of gpfdist.

Follow these steps In UNIX to stop an instance of gpfdist.

- 1 Find the process ID:

```
$ ps ax | grep gpfdist (Linux)
$ ps -ef | grep gpfdist (Solaris)
```

- 2 Kill the process. Here is an example:

```
$ kill 3456
```

## Troubleshooting gpfdist

Run this command to test connectivity between an instance of gpfdist and a HAWQ database segment.

```
$ wget http://gpfdist_hostname:port/file-name
```

## Using the file:// Protocol

You can use the file:// protocol to identify external files for bulk loading with no additional configuration required. However, using the GPOLOAD\_HOME environment variable is highly recommended. If you do not specify GPOLOAD\_HOME, the BL\_DATAFILE data set option specifies the source directory. The default source directory is the current directory if you do not specify BL\_DATAFILE=. The HAWQ server must have access to the source directory.

## Accessing Dynamic Data in Web Tables

Use these data set options to access web tables:

- **BL\_EXECUTE\_CMD=**

- [BL\\_LOCATION=](#)

## Data Set Options for Bulk Loading

Here are the HAWQ bulk-load data set options. For detailed information about these options, see [Data Set Options for Relational Databases on page 370](#).

- [BL\\_DATAFILE=](#)
- [BL\\_CLIENT\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DELIMITER=](#)
- [BL\\_ENCODING=](#)
- [BL\\_ESCAPE=](#)
- [BL\\_EXCEPTION=](#)
- [BL\\_EXECUTE\\_CMD=](#)
- [BL\\_EXECUTE\\_LOCATION=](#)
- [BL\\_EXTERNAL\\_WEB=](#)
- [BL\\_FORCE\\_NOT\\_NULL=](#)
- [BL\\_FORMAT=](#)
- [BL\\_HEADER=](#)
- [BL\\_HOST=](#)
- [BL\\_NULL=](#)
- [BL\\_PORT=](#)
- [BL\\_PROTOCOL=](#)
- [BL\\_QUOTE=](#)
- [BL\\_REJECT\\_LIMIT=](#)
- [BL\\_REJECT\\_TYPE=](#)
- [BL\\_USE\\_PIPE=](#)
- [BULKLOAD=](#)

## Bulk Loading Examples

This first example shows how you can use a SAS data set, SASFLT.FLT98, to create and load a HAWQ table, FLIGHTS98.

```
libname sasflt 'SAS-data-library';
libname mydblib hawq server=mysrv1_users
      db=users user=myusr1 password=mypwd1;

proc sql;
```

```

create table net_air.flights98
(BULKLOAD=YES
BL_DATAFILE='c:\temp\HAWQ\data.dat'
BL_USE_PIPE=NO
BL_DELETE_DATAFILE=yes
BL_HOST='192.168.x.x'
BL_PORT=8081)
as select * from sasflt.flt98;
quit;

```

This next example shows how you can append the SAS data set, SASFLT.FLT98, to the existing HAWQ table ALLFLIGHTS. The BL\_USE\_PIPE=NO option forces SAS/ACCESS Interface to HAWQ to write data to a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the load has completed.

```

proc append base=new_air.flights98
(BULKLOAD=YES
BL_DATAFILE='c:\temp\HAWQ\data.dat'
BL_USE_PIPE=NO
BL_DELETE_DATAFILE=no
BL_HOST='192.168.x.x'
BL_PORT=8081)
data=sasflt.flt98;
run;

```

## Locking in the HAWQ Interface

These LIBNAME and data set options let you control how the HAWQ interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134. For additional information, see your HAWQ documentation.

**READ\_LOCK\_TYPE**= ROW (default) | TABLE

**UPDATE\_LOCK\_TYPE**= ROW (default) | TABLE

**READ\_ISOLATION\_LEVEL**= RC | RR | RU | S | V

The HAWQ database manager supports the RC, RR, RU, S, and V isolation levels that are defined in the following table. The default value is RC.

Regardless of the isolation level, the database manager places exclusive locks on every row that is inserted, updated, or deleted. All isolation levels therefore ensure that only this application process can change any given row during a unit of work. No other application process can change any rows until the unit of work is complete.

**Table 23.3** Isolation Levels for HAWQ

| Isolation Level       | Definition                                                                                                                                                                                                                                               |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RC (read uncommitted) | Allows dirty read, nonrepeatable read, and phantom Read operations.                                                                                                                                                                                      |
| RR (repeatable read)  | Does not allow dirty read, nonrepeatable read, or phantom Read operations.                                                                                                                                                                               |
| RU (read uncommitted) | Allows dirty read, nonrepeatable read, and phantom Read operations.                                                                                                                                                                                      |
| S (serializable)      | Does not allow dirty read, nonrepeatable read, or phantom Read operations.                                                                                                                                                                               |
| V (versioning)        | Does not allow dirty read, nonrepeatable read, or phantom Read operations. These transactions are serializable, but higher concurrency is possible with this level than with the serializable isolation level. Typically, a nonlocking protocol is used. |

Here is how the terms in the table are defined.

#### *Dirty reads*

A transaction that has very minimal isolation from concurrent transactions. In fact, this type of transaction can see changes that concurrent transactions make even before those transactions commit their changes.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

#### *Nonrepeatable reads*

A transaction where the system reads a row once and is unable to read the row again later in the same transaction. This might occur if a concurrent transaction changed or even deleted the row. Therefore, the Read operation is not (necessarily) repeatable.

For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

#### *Phantom reads*

A transaction where a set of rows that is read once might become a different set of rows if the transaction attempts to read the rows again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy a condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist (a “phantom”).

`UPDATE_ISOLATION_LEVEL= RC | RR | S | V`

The HAWQ database manager supports the RC, RR, S, and V isolation levels that are in the preceding table. Uncommitted reads are not allowed with this option. The default is RC.

---

# Naming Conventions for HAWQ

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21..](#)

Most SAS names can be up to 32 characters long. The HAWQ interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name results in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The `PRESERVE_TAB_NAMES=` and `PRESERVE_COL_NAMES=` options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) HAWQ is not case sensitive, so all names default to lowercase.

HAWQ objects include tables, views, and columns. They follow these naming conventions.

- A name can contain up to 128 characters.
- The first character in a name can be a letter, @, \_, or #.
- A name cannot be a HAWQ reserved word, such as WHERE or VIEW.
- A name must be unique within each type of each object.

---

# Data Types for HAWQ

---

## Overview

Every column in a table has a name and a data type. The data type tells HAWQ how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about HAWQ data types, null and default values, and data conversions.

SAS/ACCESS Interface to HAWQ does not directly support any data types that are not listed below. Any columns using these types are read into SAS as character strings.

For more information about HAWQ data types and to determine which data types are available for your version of HAWQ, see your HAWQ documentation.

---

## Supported HAWQ Data Types

Here are the data types that SAS/ACCESS Interface to HAWQ supports:

- Character data:

CHAR(*n*)  
VARCHAR(*n*)  
TEXT

- Numeric data:

|                  |                         |
|------------------|-------------------------|
| BIGINT           | REAL                    |
| SMALLINT         | FLOAT                   |
| INTEGER          | DECIMAL   DEC   NUMERIC |
| DOUBLE PRECISION |                         |

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

---

- Date, time, and timestamp data:

DATE  
TIME  
TIMESTAMP

---

**Note:** Be aware that columns of these data types can contain data values that are out of range for SAS.

---

- Binary data: BYTEA

---

## HAWQ Null Values

HAWQ has a special value called NULL. A HAWQ NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a HAWQ NULL value, it interprets it as a SAS missing value. When loading SAS tables from HAWQ sources, SAS/ACCESS stores HAWQ NULL values as SAS missing values.

In HAWQ tables, NULL values are valid in all columns by default. There are two methods to specify a column in a HAWQ table so that it requires data:

- Using SQL, you specify a column as NOT NULL. This tells SQL to allow only a row to be added to a table if a value exists for the field. Rows that contain NULL values in that column are not added to the table.
- Another approach is to assert NOT NULL DEFAULT.

When creating HAWQ tables with SAS/ACCESS, you can use the DBNULL= data set option to specify the treatment of NULL values. For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

Once you know whether a HAWQ column enables NULLs or the host system provides a default value for a column that is specified as NOT NULL WITH DEFAULT, you can write selection criteria and enter values to update a table. Unless a column is specified as NOT NULL or NOT NULL DEFAULT, it enables NULL values.

To control how the DBMS handles SAS missing character values, use the **NULLCHAR=** and **NULLCHARVAL=** data set options.

## LIBNAME Statement Data Conversions

The following table shows the default formats that SAS/ACCESS Interface to HAWQ assigns to SAS variables when using the **LIBNAME statement** to read from a HAWQ table.

These default formats are based on HAWQ column attributes.

**Table 23.4 LIBNAME Statement: Default SAS Formats for HAWQ Data Types**

| HAWQ Data Type                     | SAS Data Type | Default SAS Format       |
|------------------------------------|---------------|--------------------------|
| CHAR( <i>n</i> ) <sup>1</sup>      | character     | \$ <i>w</i> .            |
| VARCHAR( <i>n</i> ) <sup>1</sup>   | character     | \$ <i>w</i> .            |
| BYTEA                              | character     | \$ <i>w</i> <sup>3</sup> |
| INTEGER                            | numeric       | 11.                      |
| SMALLINT                           | numeric       | 6.                       |
| BIGINT                             | numeric       | 20.                      |
| DECIMAL( <i>p,s</i> ) <sup>2</sup> | numeric       | <i>w.d</i>               |
| NUMERIC( <i>p,s</i> ) <sup>2</sup> | numeric       | <i>w.d</i>               |
| REAL                               | numeric       | none                     |
| TIME                               | numeric       | TIME8.                   |
| DATE                               | numeric       | DATE9.                   |

| HAWQ Data Type | SAS Data Type | Default SAS Format |
|----------------|---------------|--------------------|
| TIMESTAMP      | numeric       | DATETIME25.6       |

- 1  $n$  in HAWQ data types is equivalent to  $w$  in SAS formats.
- 2  $p$  and  $s$  in HAWQ numeric data types are equivalent to  $w$  and  $d$  in SAS formats.
- 3 Because the Greenplum ODBC driver does the conversion, this field is displayed as if the \$HEXw. format were applied.

The next table shows the default HAWQ data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

*Table 23.5 LIBNAME Statement: Default HAWQ Data Types for SAS Variable Formats*

| SAS Variable Format | HAWQ Data Type                |
|---------------------|-------------------------------|
| $w.d$               | DECIMAL( $p,s$ ) <sup>2</sup> |
| $\$w.$              | VARCHAR( $n$ ) <sup>1</sup>   |
| datetime formats    | TIMESTAMP                     |
| date formats        | DATE                          |
| time formats        | TIME                          |

- 1  $n$  in HAWQ data types is equivalent to  $w$  in SAS formats.
- 2  $p$  and  $s$  in HAWQ numeric data types are equivalent to  $w$  and  $d$  in SAS formats.

## Sample Programs for HAWQ

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub repository](#).

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to Impala

|                                                                      |            |
|----------------------------------------------------------------------|------------|
| <b><i>System Requirements for SAS/ACCESS Interface to Impala</i></b> | <b>952</b> |
| <b><i>Introduction to SAS/ACCESS Interface to Impala</i></b>         | <b>952</b> |
| Overview                                                             | 952        |
| Impala Concepts                                                      | 952        |
| <b><i>LIBNAME Statement for the Impala Engine</i></b>                | <b>953</b> |
| Overview                                                             | 953        |
| Arguments                                                            | 953        |
| Impala LIBNAME Statement Examples                                    | 957        |
| <b><i>Data Set Options for Impala</i></b>                            | <b>957</b> |
| <b><i>SQL Pass-Through Facility Specifics for Impala</i></b>         | <b>959</b> |
| Key Information                                                      | 959        |
| CONNECT Statement Examples                                           | 960        |
| <b><i>Passing SAS Functions to Impala</i></b>                        | <b>960</b> |
| <b><i>Passing Joins to Impala</i></b>                                | <b>961</b> |
| <b><i>Sorting Data That Contains NULL Values</i></b>                 | <b>961</b> |
| <b><i>Bulk Loading for Impala</i></b>                                | <b>962</b> |
| Loading                                                              | 962        |
| Data Set Options for Bulk Loading                                    | 963        |
| Configuration Details                                                | 963        |
| Example 1: Create an Impala Table from a SAS Data Set                | 964        |
| Example 2: Append a SAS Data Set to an Existing Impala Table         | 964        |
| Example 3: Create an Impala Table from a SAS Data Set                | 965        |
| <b><i>Naming Conventions for Impala</i></b>                          | <b>965</b> |
| <b><i>Data Types for Impala</i></b>                                  | <b>966</b> |
| Overview                                                             | 966        |
| Supported Impala Data Types                                          | 966        |
| SAS Data Types                                                       | 967        |
| Data Conversion from Impala to SAS                                   | 967        |
| Issues When Converting from Impala to SAS                            | 968        |
| Data Conversion from SAS to Impala                                   | 968        |
| Impala Null Values                                                   | 969        |
| <b><i>Sample Programs for Impala</i></b>                             | <b>969</b> |

---

# System Requirements for SAS/ACCESS Interface to Impala

You can find information about system requirements for SAS/ACCESS Interface to Impala in these locations.

- [System Requirements for SAS/ACCESS Interface to Impala with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to Impala

---

## Overview

For available SAS/ACCESS features, see [Impala supported features on page 105](#). For more information about Impala, see your Impala documentation.

---

**Note:** The SAS/ACCESS Interface to Impala was implemented in [SAS 9.4M2](#).

---

Beginning in [SAS Viya 3.3](#), SAS/ACCESS Interface to Impala includes SAS Data Connector to Impala. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“Impala Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

---

## Impala Concepts

Cloudera Impala is an open-source, massively parallel processing (MPP) query engine that runs natively on Apache Hadoop. You can use it to issue SQL queries to data stored in HDFS and Apache Hbase without moving or transforming data.

Similar to other SAS/ACCESS engines, SAS/ACCESS Interface to Impala lets you run SAS procedures against data that is stored in Impala and returns the results to SAS. You can use it to read and write data to and from Hadoop as if it were any other relational data source to which SAS can connect.

---

# LIBNAME Statement for the Impala Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Impala supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Impala.

**LIBNAME** *libref* **impala** <*connection-options*> <*LIBNAME-options*>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in *SAS Global Statements: Reference*.

---

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *impala*

specifies the SAS/ACCESS engine name for the Impala interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are specified. When you use the LIBNAME statement, you can connect to the Impala database in two ways. Specify only one of these methods for each connection because they are mutually exclusive.

- SERVER=, DATABASE=, PORT=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD=

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**SERVER=<'>*Impala-server-name*<'>**

specifies the Impala server name that runs the Impala daemon. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: HOST=

**DATABASE=<'>*database-name*<'>**

specifies the Impala database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORt=*port***

specifies the port number that is used to connect to the specified Impala server.

Default: 21050

**USER=<'>*Impala-user-name*<'>**

specifies the user name.

**PASSWORD=<'>*Impala-password*<'>**

specifies the Impala password that is associated with your user ID. If it contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. If you do not want to enter your Impala password in uncoded text on this statement, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

Alias: PASS=, PWD=, PW=

**DATASRC=<'>*Impala-data-source*<'>**

specifies the configured Impala ODBC data source to which you want to connect. It is recommended that you use this option only if you have configured Impala ODBC data sources on your client. This method requires additional setup—either through the ODBC Administrator control panel on Windows platforms, or through the odbc.ini file or a similarly named configuration file on UNIX platforms. It is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

Aliases: DS=, DSN=

**SCHEMA=*Impala-schema***

specifies the Impala schema.

Alias: DATABASE=, DB=

Default: none

**HDFS\_TEMPDIR='path'**

specifies the path to the HDFS directory where SAS reads and writes temporary data.

Default: HDFS\_TEMPDIR=' /tmp '

**CONOPTS=<'> Impala ODBC-connections-options<'>**

specifies connection options for your data source or database. Separate multiple options with a semicolon. Refer to your data source or database's ODBC driver documentation for a list of the ODBC connection options that your ODBC driver supports.

Required: The minimum Impala ODBC driver is 2.5.29.

*LIBNAME-options*

specify how SAS processes DBMS objects. The following table describes the LIBNAME options for SAS/ACCESS Interface to Impala, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For more information, see [LIBNAME Options for Relational Databases on page 134](#) or [“SQL Pass-Through Facility” on page 689](#).

**Table 24.1** SAS/ACCESS LIBNAME Options for Impala

| Option               | Default Value                                                                        | Valid in CONNECT |
|----------------------|--------------------------------------------------------------------------------------|------------------|
| ACCESS=              | none                                                                                 |                  |
| AUTHDOMAIN=          | none                                                                                 |                  |
| BL_HOST=             | SERVER= value, if SERVER= is specified; otherwise, none                              |                  |
| BL_PORT=             | 50070 (required only if Impala HDFS streaming is running on a nondefault port)       |                  |
| BULKLOAD=            | none                                                                                 |                  |
| CONNECTION=          | SHAREDREAD                                                                           | •                |
| CONNECTION_GROUP=    | none                                                                                 | •                |
| CONOPTS=             | none                                                                                 | •                |
| CURSOR_TYPE=         | FORWARD_ONLY                                                                         | •                |
| DBCLIENT_MAX_BYTES=  | matches the maximum number of bytes per single character of the SAS session encoding | •                |
| DBCOMMIT=            | none                                                                                 |                  |
| DBCONINIT=           | none                                                                                 | •                |
| DBCONTERM=           | none                                                                                 | •                |
| DBCREATE_TABLE_OPTS= | none                                                                                 | •                |
| DBGEN_NAME=          | DBMS                                                                                 | •                |
| DBINDEX=             | NO                                                                                   |                  |

| Option              | Default Value                                | Valid in CONNECT |
|---------------------|----------------------------------------------|------------------|
| DBMAX_TEXT=         | 1024                                         | •                |
| DBNULLKEYS=         | YES                                          |                  |
| DBPROMPT=           | NO                                           | •                |
| DBSASLABEL=         | COMPAT                                       |                  |
| DBSERVER_MAX_BYTES= | 0                                            | •                |
| DEFER=              | NO                                           | •                |
| DELETE_MULT_ROWS=   | NO                                           |                  |
| DIRECT_SQL=         | YES                                          |                  |
| DRIVER_VENDOR=      | CLOUDERA                                     | •                |
| HDFS_PRINCIPAL=     | none                                         | •                |
| INSERTBUFF=         | automatically calculated based on row length |                  |
| IMPALA_PRINCIPAL=   | none                                         | •                |
| LOGIN_TIMEOUT=      | 0                                            | •                |
| MULTI_DATASRC_OPT=  | NONE                                         |                  |
| POST_STMT_OPTS=     | none                                         |                  |
| PRESERVE_COL_NAMES= | YES                                          |                  |
| PRESERVE_TAB_NAMES= | YES                                          |                  |
| QUERY_TIMEOUT=      | 0                                            | •                |
| READBUFF=           | automatically calculated based on row length | •                |
| REREAD_EXPOSURE=    | NO                                           | •                |
| SCHEMA=             | none                                         | •                |
| SCRATCH_DB=         | none                                         |                  |
| SPOOL=              | YES                                          |                  |

| Option              | Default Value | Valid in CONNECT |
|---------------------|---------------|------------------|
| SQL_FUNCTIONS=      | none          |                  |
| SQL_FUNCTIONS_COPY= | none          |                  |
| SQLGENERATION=      | none          |                  |
| STRINGDATES=        | NO            | •                |
| SUB_CHAR=           | none          |                  |
| TRACE=              | NO            | •                |
| TRACEFILE=          | none          | •                |
| UPDATE_MULT_ROWS=   | NO            |                  |
| USE_DATADIRECT=     | NO            | •                |

---

## Impala LIBNAME Statement Examples

This example uses the default Impala port.

```
libname imp impala server=hxduped schema=myschema
      user=myusr1 password=mypwd1;
```

This example explicitly specifies the default Impala port.

```
libname imp impala server=hxduped port=21050 schema=myschema
      user=myusr1 password=mypwd1;
```

---

## Data Set Options for Impala

All SAS/ACCESS data set options in this table are supported for Impala. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 24.2** SAS/ACCESS Data Set Options for Impala

| Option       | Default Value           |
|--------------|-------------------------|
| BL_DATAFILE= | automatically generated |

| Option                   | Default Value                                                                                                                             |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| BL_DELETE_DATAFILE=      | YES                                                                                                                                       |
| BL_HOST=                 | SERVER= value, if SERVER= is specified; otherwise, none                                                                                   |
| BL_PORT=                 | 50070                                                                                                                                     |
| BULKLOAD=                | none                                                                                                                                      |
| CURSOR_TYPE=             | LIBNAME option value                                                                                                                      |
| DBCOMMIT=                | LIBNAME option value                                                                                                                      |
| DBCONDITION=             | none                                                                                                                                      |
| DBCREATE_TABLE_EXTERNAL= | NO                                                                                                                                        |
| DBCREATE_TABLE_LOCATION= | /user/hive/warehouse/ <i>tabname</i> [with the default schema], /user/hive/warehouse/ <i>schema.db/tabname</i> [with a nondefault schema] |
| DBCREATE_TABLE_OPTS=     | LIBNAME option value                                                                                                                      |
| DBFORCE=                 | NO                                                                                                                                        |
| DBGEN_NAME=              | DBMS                                                                                                                                      |
| DBINDEX=                 | NO                                                                                                                                        |
| DBLABEL=                 | NO                                                                                                                                        |
| DBLARGETABLE=            | none                                                                                                                                      |
| DBMAX_TEXT=              | 1024                                                                                                                                      |
| DBNULLKEYS=              | LIBNAME option value                                                                                                                      |
| DBPROMPT=                | LIBNAME option value                                                                                                                      |
| DBSASLABEL=              | COMPAT                                                                                                                                    |
| DBSASTYPE=               | see Data Types for Impala                                                                                                                 |
| DBTYPE=                  | see Data Types for Impala                                                                                                                 |
| HDFS_PRINCIPAL=          | none                                                                                                                                      |

| Option              | Default Value        |
|---------------------|----------------------|
| INSERTBUFF=         | LIBNAME option value |
| PARTITIONED_BY=     | none                 |
| POST_STMT_OPTS=     | none                 |
| POST_TABLE_OPTS=    | none                 |
| PRE_STMT_OPTS=      | none                 |
| PRE_TABLE_OPTS=     | none                 |
| PRESERVE_COL_NAMES= | LIBNAME option value |
| QUERY_TIMEOUT=      | LIBNAME option value |
| READBUFF=           | LIBNAME option value |
| SCHEMA=             | LIBNAME option value |
| SCRATCH_DB=         | none                 |
| SUB_CHAR=           | none                 |

---

## SQL Pass-Through Facility Specifics for Impala

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Impala interface.

- The *dbms-name* is **IMPALA**.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Impala. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default **IMPALA** alias is used.

- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#). Additional LIBNAME options that are valid in the CONNECT statement are indicated in [Table 24.1 on page 955](#).

## CONNECT Statement Examples

This example uses the default Impala port.

```
proc sql;
  connect to impala (user="myusr1" pw="mypwd1" server=hxduped) schema=myschema;
```

This example explicitly specifies the default Impala port.

```
proc sql;
  connect to impala (user="myusr1" pw="mypwd1" server=hxduped
  port=21050 schema=myschema schema=default);
```

## Passing SAS Functions to Impala

SAS/ACCESS Interface to Impala passes the following SAS functions to Impala for processing. Where the Impala function name differs from the SAS function name, the Impala name appears in parentheses. For more information, see [“Passing Functions to the DBMS Using PROC SQL” on page 58](#).

|                |                          |
|----------------|--------------------------|
| ** (POWER)     | LOG10                    |
| ABS            | LOWCASE (LOWER)          |
| ARCOS (ACOS)   | MAX                      |
| ARSIN (ASIN)   | MIN                      |
| ATAN           | MINUTE                   |
| AVG            | MONTH                    |
| CAT (CONCAT)   | SECOND                   |
| CEIL           | SIN                      |
| COS            | SQRT                     |
| COUNT          | STRIP (TRIM)             |
| DAY            | SUBSTR                   |
| EXP            | SUM                      |
| FLOOR          | TAN                      |
| HOUR           | TRANWRD (REGEXP_REPLACE) |
| INDEX (LOCATE) | TRIMN (RTRIM)            |
| LEFT (LTRIM)   | UPCASE (UPPER)           |
| LENGTH         | YEAR                     |
| LOG (LN)       |                          |

[SQL\\_FUNCTIONS=ALL](#) allows for SAS functions that have slightly different behavior from corresponding Impala functions that are passed down to Impala. Only

when SQL\_FUNCTIONS=ALL can the SAS/ACCESS engine also pass these SAS SQL functions to Impala. Due to incompatibility in date and time functions between Impala and SAS, Impala might not process them correctly. Check your results to determine whether these functions are working as expected.

|                           |                    |
|---------------------------|--------------------|
| COMPRESS (REGEXP_REPLACE) | DATETIME (NOW())   |
| DATE (CURRDATE())         | TODAY (CURRDATE()) |
| DATEPART (TO_DATE())      |                    |

---

## Passing Joins to Impala

In order for a multiple-libref join to be passed to the database, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)
- port (PORT=)
- schema (SCHEMA=)
- server (SERVER=)
- data source (DSN=, if specified)

You can use the [SQL pass-through facility](#) to pass a cross-schema join to your database. For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Sorting Data That Contains NULL Values

Some DBMSs, including Impala, place NULL values at the end of a sorted list. SAS normally places NULL values at the beginning of a sorted list. In SAS, to use BY-group processing, SAS expects that values for a BY group are already sorted before it performs BY-group processing. If SAS determines that values are not sorted, then BY-group processing stops. Therefore, if NULL values are not at the beginning of a sorted list, SAS determines that the values are not sorted correctly.

If you use PROC SQL with an ORDER BY statement to sort your data, then the SQL code is generated so that NULL values are placed at the beginning of the sorted list. In this way, SAS can perform any BY-group processing without stopping.

For more information, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43 and “[Sorting DBMS Data](#)” on page 49.

---

# Bulk Loading for Impala

---

## Loading

Bulk loading with the Impala engine is accomplished in two ways:

- Use the WebHDFS interface to Hadoop to push data to HDFS. The SAS environment variable SAS\_HADOOP\_RESTFUL must be specified and set to the value of 1. You can include the properties for the WebHDFS location in the Hadoop hdfs-site.xml file. Alternatively, specify the WebHDFS host name or the IP address of the server where the external file is stored using the BL\_HOST= option. The BULKLOAD= option must be set to YES. No JAR files are needed.

**Note:** Support for the SAS\_HADOOP\_RESTFUL environment variable was added in SAS 9.4M2.

- Configure a required set of Hadoop JAR files. JAR files must be in a single location and available to the SAS client machine. The SAS environment variable SAS\_HADOOP\_JAR\_PATH must be specified and set to the location of the Hadoop JAR files. Refer to the *SAS 9.4 Hadoop Configuration Guide for Base SAS and SAS/ACCESS* for information about what Cloudera JAR files are required.

**Note:** Support for configuration using Hadoop JAR files and the SAS\_HADOOP\_JAR\_PATH environment variable was added in SAS 9.4M2.

Here is how the Impala engine creates table data using the bulk-loading process:

- 1 SAS issues two CREATE TABLE statements to the Impala server. One CREATE TABLE statement creates the target Impala table. The other CREATE TABLE statement creates a temporary table.
- 2 SAS uses WebHDFS or Java to upload table data to the HDFS temporary directory, as specified in HDFS\_TEMPDIR=, where /tmp is the default directory. The resulting file is a UTF-8 delimited text file.
- 3 SAS issues a LOAD DATA statement to move the data file from the HDFS /tmp directory into the temporary table.
- 4 SAS issues an INSERT INTO statement that copies and transforms the text data from the temporary table into the target Impala table.
- 5 SAS deletes the temporary table.

---

## Data Set Options for Bulk Loading

Here are the Impala bulk-load data set options. The BULKLOAD= data set option is required for bulk loading, and all others are optional. For more information, see “[Data Set Options](#)” on page 365.

- [BL\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_HOST=](#)
- [BL\\_PORT=](#)
- [BULKLOAD=](#)
- [HDFS\\_PRINCIPAL=](#)

---

## Configuration Details

Java JAR files, HDFS host-specific configuration information, or both are required to upload table data to HDFS. The host that is required for HDFS uploads when you are using Java or WebHDFS might differ from the Impala host. For example, the host might be another machine on the same cluster. Depending on your configuration, you must specify this information when you are bulk loading data to Impala.

Here is what to specify if you use WEBHDFS to upload table data.

- [SAS\\_HADOOP\\_RESTFUL=1](#) (required).
- No SAS Hadoop JAR path is required when you use WebHDFS to upload table data.
- [SAS\\_HADOOP\\_CONFIG\\_PATH=<config-directory>](#) (contains the hdfs-site.xml file for the specific HDFS host). As an alternative, you can use the BL\_HOST data set option to specify the HDFS host name ([BL\\_HOST="hdfshost"](#)). Using [SAS\\_HADOOP\\_CONFIG\\_PATH=](#) is the preferred solution.

Here is what to specify if you use Java to upload table data.

- [SAS\\_HADOOP\\_RESTFUL=0](#) (optional).
- [SAS\\_HADOOP\\_JAR\\_PATH=<jar-directory>](#).
- [SAS\\_HADOOP\\_CONFIG\\_PATH=<config-directory>](#) (contains the hdfs-site.xml file for the specific host). As an alternative, you can use the BL\_HOST data set option to specify the HDFS host name ([BL\\_HOST="hdfshost"](#)). Using [SAS\\_HADOOP\\_CONFIG\\_PATH=](#) is the preferred solution.

[SAS\\_HADOOP\\_RESTFUL](#), [SAS\\_HADOOP\\_JAR\\_PATH](#), and [SAS\\_HADOOP\\_CONFIG\\_PATH](#) are environment variables. Here is how you can specify them:

- as an operating system environment variable before starting SAS
- as a SAS -SET option, such as `SAS ... -SET SAS_HADOOP_RESTFUL 1`

- as an OPTION SET command within SAS, such as option  
set=SAS\_HADOOP\_RESTFUL 1;

## Example 1: Create an Impala Table from a SAS Data Set

This example shows how you can use a SAS data set, SASFLT.FLT98, to create and load an Impala table, FLIGHTS98.

```
libname sasflt 'SAS-data-library';
libname mydblib impala host=mysrv1
    db=users user=myusr1 password=mypwd1;

proc sql;
create table mydblib.flights98
(BULKLOAD=YES
BL_DATAFILE='/tmp/mytable.dat'
BL_HOST='192.168.x.x'
BL_PORT=50070)
as select * from sasflt.flt98;
quit;
```

## Example 2: Append a SAS Data Set to an Existing Impala Table

This example shows how you can append the SAS data set, SASFLT.FLT98, to the existing Impala table, FLIGHTS98. In this example, the HDFS\_PRINCIPAL data set option is specified as well. You specify the HDFS\_PRINCIPAL= data set option when you configure HDFS to allow Kerberos authentication. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the load has completed.

```
proc append base=mydblib.flights98
(BULKLOAD=yes
BL_DATAFILE='/tmp/mytable.dat'
BL_DELETE_DATAFILE=no
HDFS_PRINCIPAL='hdfs/hdfs_host.example.com@test.example.com'
BL_HOST='192.168.x.x'
BL_PORT=50070)
data=sasflt.flt98;
run;
```

---

## Example 3: Create an Impala Table from a SAS Data Set

This example shows how to use a SAS data set, SASFLT.FLT98, to create and load an Impala table, FLIGHTS98, using WebHDFS and configuration files.

```
option set=SAS_HADOOP_RESTFUL 1;
option set=SAS_HADOOP_CONFIG_PATH "/configs/myhdfshost";
/* This path should point to the directory that contains */
/* the hdfs-site.xml file for the cluster that hosts the */
/* 'mysrv1' Impala host */

libname sasflt 'SAS-data-library';
libname mydblib impala host=mysrv1
    db=users user=myusrl password=mypwd1;

proc sql;
create table mydblib.flights98
(BULKLOAD=YES
BL_DATAFILE='/tmp/mytable.dat'
/* no BL_HOST or BL_PORT */
as select * from sasflt.flt98;
quit;
```

---

## Naming Conventions for Impala

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Impala interface supports table names and column names that contain up to 32 characters. If column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical column names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a name longer than 32 characters. If you have a table name that is greater than 32 characters, it is recommended that you create a table view.

Although the PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options are supported for SAS/ACCESS Interface to Impala, you should not need to use them. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Impala is not case sensitive, so all names default to lowercase.

SAS and Impala objects include tables, views, table references, and columns. They follow these naming conventions.

- A SAS name must be from 1 to 32 characters long. When Impala column names and table names are 32 characters or less, SAS handles them seamlessly. When SAS reads Impala column names that are longer than 32 characters, a generated SAS variable name is truncated to 32 characters. Impala table names should be 32 characters or less because SAS cannot truncate a table reference. If you already have a table name that is greater than 32 characters, create an Impala table view or use the explicit SQL feature of PROC SQL to access the table.
- If truncating would result in identical names, SAS generates a unique name.
- Even when it is enclosed in single or double quotation marks, an Impala name does not retain case sensitivity. Impala table and column names can contain uppercase letters A through Z (A–Z), lowercase letters A through Z (a–z), numbers from 0 to 9, and the underscore (\_). Impala converts uppercase characters to lowercase. Therefore, such SAS table references as MYTAB and mytab are synonyms that refer to the same table.
- A name can begin with a letter but not an underscore or a number.
- A name cannot be an Impala reserved word. If a name generates an Impala error, try to append a number or underscore in an appropriate place. For example, if *shipped* results in an error, try *shipped1* or *ship\_date*.

Although the PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options are supported for SAS/ACCESS Interface to Impala, you should not need to use them. (For information about these options, see [LIBNAME Option for Relational Databases on page 127](#).)

## Data Types for Impala

### Overview

Every column in a table has a name and a data type. The data type tells Impala how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Impala data types, null and default values, and data conversions.

SAS/ACCESS Interface to Impala does not directly support any data types that are not listed below. Any columns that use unsupported data types are read into SAS as character strings.

For more information about Impala data types and to determine which data types are available for your version of Impala, see your Impala documentation.

## Supported Impala Data Types

Here are the data types that the Impala engine supports.

- Numeric data:

|           |          |
|-----------|----------|
| BIGINT    | FLOAT    |
| TIMESTAMP | INT      |
| BOOLEAN   | SMALLINT |
| DOUBLE    | TINYINT  |
| DECIMAL   |          |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Character data:

|          |             |
|----------|-------------|
| CHAR $n$ | VARCHAR $n$ |
| STRING   |             |

- Date and time data: TIMESTAMP

**Note:** In [SAS 9.4M3](#), support for CHAR and VARCHAR data types was added.

To use CHAR and VARCHAR, an Impala 2.0 server and the Cloudera 2.5.22 or higher ODBC client driver are required.

---

## SAS Data Types

SAS has two fundamental data types, character and numeric. SAS character variables (columns) are of a fixed length with a maximum of 32,767 characters. SAS numeric variables are signed eight-byte, floating-point numbers. When SAS numerics are used in conjunction with SAS formats, they can represent a number of data types, including DATE, TIME, and DATETIME. For more information about SAS data types, see [SAS Programmer’s Guide: Essentials](#).

---

## Data Conversion from Impala to SAS

This table shows the default SAS formats that are assigned to SAS variables that are created when SAS/ACCESS reads Impala table columns.

**Note:** In [SAS 9.4M3](#), support for CHAR and VARCHAR data types was added.

**Table 24.3** *Impala to SAS: Default SAS Formats for Impala Data Types*

| <b>Impala Data Type</b> | <b>SAS Data Type</b> | <b>Default SAS Format</b> |
|-------------------------|----------------------|---------------------------|
| CHAR                    | character            | \$255.                    |
| STRING                  |                      | \$32767.                  |
| VARCHAR                 |                      | \$32767                   |
| BOOLEAN                 | numeric              | 1.                        |
| BIGINT                  |                      | 20.                       |
| DOUBLE                  |                      | none                      |
| FLOAT                   |                      | none                      |
| INT                     |                      | 11.                       |
| SMALLINT                |                      | 6.                        |
| TINYINT                 |                      | 4.                        |

---

## Issues When Converting from Impala to SAS

Below are some issues that you might face when you convert from Impala to SAS:

- STRING: Depending on the length of Impala string data, the SAS character format (\$32767.) might be unnecessarily large for short STRING columns. Alternatively, the SAS character format (\$32767.) might truncate Impala STRING columns that contain more than 32,767 characters.
- BIGINT: Converting an Impala BIGINT to a SAS numeric can result in loss of precision because the internal SAS eight-byte, floating-point format accurately preserves only 15 digits of precision. An Impala BIGINT preserves up to 19 digits of precision.

---

## Data Conversion from SAS to Impala

This table shows the Impala data types that are assigned when SAS/ACCESS Interface to Impala creates an Impala table.

**Table 24.4** SAS to Impala: Default Impala Data Types for SAS Formats

| SAS Data Type         | SAS Format  | Impala Data Type       |
|-----------------------|-------------|------------------------|
| character             | \$n.        | CHAR(n)                |
|                       | \$n.        | STRING                 |
|                       | \$n.        | VARCHAR(n)             |
| numeric               | DATETIMEw.p | TIMESTAMP              |
|                       | DATEw.      | TIMESTAMP              |
|                       | TIMEw.      | TIMESTAMP <sup>1</sup> |
|                       | 1. to 2.    | TINYINT                |
|                       | 3. to 4.    | SMALLINT               |
|                       | 5. to 9.    | INT                    |
|                       | 10. to 18.  | BIGINT                 |
| other numeric formats |             | DOUBLE                 |

<sup>1</sup> Apache Impala does not support time values with no associated date. Time values with no date are passed to Impala with the CURRENT\_DATE.

## Impala Null Values

Impala has a special value called NULL. An Impala NULL value represents an absence of information and is analogous to a SAS missing value. When SAS/ACCESS Interface to Impala reads an Impala NULL value, it interprets it as a SAS missing value. For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

## Sample Programs for Impala

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be

found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to Informix

|                                                                       |            |
|-----------------------------------------------------------------------|------------|
| <b>System Requirements for SAS/ACCESS Interface to Informix .....</b> | <b>972</b> |
| <b>Introduction to SAS/ACCESS Interface to Informix .....</b>         | <b>972</b> |
| Overview .....                                                        | 972        |
| Default Environment .....                                             | 972        |
| <b>LIBNAME Statement for the Informix Engine .....</b>                | <b>973</b> |
| Overview .....                                                        | 973        |
| Arguments .....                                                       | 973        |
| Informix LIBNAME Statement Example .....                              | 976        |
| <b>Data Set Options for Informix .....</b>                            | <b>976</b> |
| <b>SQL Pass-Through Facility Specifics for Informix .....</b>         | <b>977</b> |
| Key Information .....                                                 | 977        |
| Stored Procedures and the SQL Pass-Through Facility .....             | 978        |
| Command Restrictions for the SQL Pass-Through Facility .....          | 979        |
| Examples .....                                                        | 979        |
| <b>Autopartitioning Scheme for Informix .....</b>                     | <b>980</b> |
| Overview .....                                                        | 980        |
| Autopartitioning Restrictions .....                                   | 981        |
| Using WHERE Clauses .....                                             | 981        |
| Using DBSLICEPARM= .....                                              | 981        |
| Using DBSLICE= .....                                                  | 982        |
| <b>Temporary Table Support for Informix .....</b>                     | <b>982</b> |
| <b>Passing SAS Functions to Informix .....</b>                        | <b>982</b> |
| <b>Passing Joins to Informix .....</b>                                | <b>983</b> |
| <b>Locking in the Informix Interface .....</b>                        | <b>983</b> |
| <b>Naming Conventions for Informix .....</b>                          | <b>984</b> |
| <b>Data Types for Informix .....</b>                                  | <b>985</b> |
| Overview .....                                                        | 985        |
| Supported Informix Data Types .....                                   | 985        |
| LIBNAME Statement Data Conversions .....                              | 986        |

|                                                  |            |
|--------------------------------------------------|------------|
| SQL Pass-Through Facility Data Conversions ..... | 988        |
| <b>Informix Servers .....</b>                    | <b>988</b> |
| Overview: Informix Database Servers .....        | 988        |
| Using the DBDATASRC Environment Variable .....   | 989        |
| Using Fully Qualified Table Names .....          | 989        |

---

# System Requirements for SAS/ACCESS Interface to Informix

You can find information about system requirements for SAS/ACCESS Interface to Informix in the following locations:

- [System Requirements for SAS/ACCESS Interface to Informix with SAS 9.4](#)
- [Third-Party Software Requirements for Use with SAS 9.4](#)

# Introduction to SAS/ACCESS Interface to Informix

## Overview

For available SAS/ACCESS features, see [Informix supported features on page 105](#). For background information about Informix, see [Informix servers on page 988](#). For more information about Informix, see your Informix documentation.

**Note:** SAS/ACCESS Interface to Informix is not included with SAS Viya 3.5.

## Default Environment

When you access Informix tables by using SAS/ACCESS Interface to Informix, the default Informix Read isolation level is specified for committed reads, and SAS spooling is on. Committed reads enable you to read rows unless another user or process is updating the rows. Reading in this manner does not lock the rows. SAS spooling guarantees that you obtain identical data each time you re-read a row because SAS buffers the rows after you read them the first time. This default

environment is suitable for most users. If this default environment is unsuitable for your needs, see “[Locking in the Informix Interface](#)” on page 983.

To see the SQL statements that SAS issues to the Informix server, include the **SASTRACE=** option in your code:

```
option sastrace=',,,d';
```

If you use quotation marks in your Informix SQL statements, set your **DELIMIDENT=** environment variable to **DELIMIDENT=YES** or Informix might reject your statements. Because some SAS options that preserve case generate SQL statements that contain quotation marks, you should specify **DELIMIDENT=YES** in your environment.

---

# LIBNAME Statement for the Informix Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Informix supports. For general information about this feature, see [LIBNAME Statement for Relational Databases](#) on page 127.

Here is the LIBNAME statement syntax for accessing Informix.

**LIBNAME** *libref* **informix** <connection-options> <LIBNAME-options>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

***libref***

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

***informix***

specifies the SAS/ACCESS engine name for the Informix interface.

***connection-options***

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are defined.

**USER=<'>*Informix-user-name*<'**

specifies the Informix user name that you use to connect to the database that contains the tables and views that you want to access. If you omit the **USER=** option, your operating environment account name is used, if applicable to your operating environment.

**USING=<'>Informix-password<'>**

specifies the password that is associated with the Informix user. If you omit the password, Informix uses the password in the `/etc/password` file.

Alias: `PASSWORD=`, `PWD=`

**SERVER=<'>ODBC-data-source<'>**

specifies the ODBC data source to which you want to connect. An error occurs if the `SERVER=` option is not specified. For UNIX platforms, you must configure the data source by modifying the `odbc.ini` file. See your ODBC driver documentation for details.

For the SAS/ACCESS 9 Interface to Informix, the Informix ODBC Driver API is used to connect to Informix, and connection options have changed accordingly. The `DATABASE=` option from the SAS 8 version of SAS/ACCESS was removed. If you need to specify a database, set it in the `odbc.ini` file. For `SERVER=` options, instead of specifying the server name, as in SAS 8, specify an ODBC data source name. You can also use a user ID and password with `SERVER=`.

**DBDATASRC=<'>database-data-source<'>**

environment variable that lets you specify a default data source. This value is used if you do not specify a `SERVER=` connection option.

***LIBNAME-options***

define how SAS processes DBMS objects. Some `LIBNAME` options can enhance performance, and others determine locking or naming behavior. The following table describes the `LIBNAME` options for SAS/ACCESS Interface to Informix, with the applicable default values. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 25.1** SAS/ACCESS LIBNAME Options for Informix

| Option                            | Default Value                                  |
|-----------------------------------|------------------------------------------------|
| <code>ACCESS=</code>              | none                                           |
| <code>AUTHDOMAIN=</code>          | none                                           |
| <code>AUTOCOMMIT=</code>          | YES                                            |
| <code>CONNECTION=</code>          | SHAREDREAD                                     |
| <code>CONNECTION_GROUP=</code>    | none                                           |
| <code>DBCOMMIT=</code>            | 1000 when inserting rows; 0 when updating rows |
| <code>DBCONINIT=</code>           | none                                           |
| <code>DBCONTERM=</code>           | none                                           |
| <code>DBCREATE_TABLE_OPTS=</code> | none                                           |
| <code>DBGEN_NAME=</code>          | DBMS                                           |

| <b>Option</b>         | <b>Default Value</b>                                     |
|-----------------------|----------------------------------------------------------|
| DBINDEX=              | NO                                                       |
| DBLIBINIT=            | none                                                     |
| DBLIBTERM=            | none                                                     |
| DBNULLKEYS=           | NO                                                       |
| DBPROMPT=             | NO                                                       |
| DBSASLABEL=           | COMPAT                                                   |
| DBSLICEPARM=          | THREADED_APPS,2 or<br>THREADED_APPS,3                    |
| DEFER=                | NO                                                       |
| DIRECT_EXE=           | none                                                     |
| DIRECT_SQL=           | YES                                                      |
| LOCKTABLE=            | no locking                                               |
| LOCKTIME=             | none                                                     |
| LOCKWAIT=             | not specified                                            |
| MULTI_DATASRC_OPT=    | NONE                                                     |
| POST_STMT_OPTS=       | none                                                     |
| PRESERVE_COL_NAMES=   | NO                                                       |
| PRESERVE_TAB_NAMES=   | NO                                                       |
| READ_ISOLATION_LEVEL= | COMMITTED READ (see “Locking in the Informix Interface”) |
| REREAD_EXPOSURE=      | NO                                                       |
| SCHEMA=               | your user name                                           |
| SPOOL=                | YES                                                      |
| SQL_FUNCTIONS=        | none                                                     |
| UTILCONN_TRANSIENT=   | NO                                                       |

## Informix LIBNAME Statement Example

In this example, the libref MYDBLIB uses the Informix interface to connect to an Informix database:

```
libname mydblib informix user=myusr1 using=mpwd1 server=mysrv1;
```

USER=, USING=, and SERVER= are connection options in this example.

## Data Set Options for Informix

All SAS/ACCESS data set options in this table are supported for Informix. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

*Table 25.2 SAS/ACCESS Data Set Options for Informix*

| Option               | Default Value                                   |
|----------------------|-------------------------------------------------|
| DBCOMMIT=            | LIBNAME option value                            |
| DBCONDITION=         | none                                            |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                            |
| DBFORCE=             | NO                                              |
| DBGEN_NAME=          | DBMS                                            |
| DBINDEX=             | LIBNAME option value                            |
| DBKEY=               | none                                            |
| DBLABEL=             | NO                                              |
| DBLARGETABLE=        | none                                            |
| DBNULL=              | _ALL_=YES                                       |
| DBNULLKEYS=          | LIBNAME option value                            |
| DBSASLABEL=          | COMPAT                                          |
| DBSASTYPE=           | see “ <a href="#">Data Types for Informix</a> ” |

| Option              | Default Value                 |
|---------------------|-------------------------------|
| DBSLICE=            | none                          |
| DBSLICEPARM=        | THREADED_APPS,2 or 3          |
| DBSLICEPARM=        | see “Data Types for Informix” |
| ERRLIMIT=           | 1                             |
| LOCKTABLE=          | LIBNAME option value          |
| NULLCHAR=           | SAS                           |
| NULLCHARVAL=        | a blank character             |
| POST_STMT_OPTS=     | none                          |
| POST_TABLE_OPTS=    | none                          |
| PRE_STMT_OPTS=      | none                          |
| PRE_TABLE_OPTS=     | none                          |
| PRESERVE_COL_NAMES= | LIBNAME option value          |
| SASDATEFMT=         | DATETIME                      |
| SCHEMA=             | LIBNAME option value          |

---

## SQL Pass-Through Facility Specifics for Informix

---

### Key Information

For general information about this feature, see “SQL Pass-Through Facility” on page 690.

Here are the SQL pass-through facility specifics for the Informix interface.

- The *dbms-name* is **informix**.

- The CONNECT statement is optional when you are connecting to an Informix database if the DBDATASRC environment variable has been specified. When you omit a CONNECT statement, an implicit connection is performed when the first EXECUTE statement or CONNECTION TO component is passed to the DBMS.
- You can connect to only one Informix database at a time. However, you can specify multiple CONNECT statements if they all connect to the same Informix database. If you use multiple connections, you must use an *alias* to identify the different connections. If you omit an alias, `informix` is automatically used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).
- If you use quotation marks in your Informix pass-through statements, your DELIMIDENT= environment variable must be set to DELIMIDENT=YES, or your statements are rejected by Informix.
- The SCHEMA LIBNAME option is ignored when using implicit pass-through.
- The SCHEMA data set option disables implicit pass-through.

## Stored Procedures and the SQL Pass-Through Facility

The SQL pass-through facility recognizes two types of stored procedures in Informix that perform only database functions. The methods for executing the two types of stored procedures are different.

### Procedures that return no values to the calling application

Stored procedures that do not return values can be executed directly by using the Informix SQL EXECUTE statement. Stored procedure execution is initiated with the Informix EXECUTE PROCEDURE statement. The following example executes the stored procedure `make_table`. The stored procedure has no input parameters and returns no values.

```
execute (execute procedure make_table())
       by informix;
```

### Procedures that return values to the calling application

Stored procedures that return values must be executed by using the PROC SQL SELECT statement with a CONNECTION TO component. This example executes the stored procedure `read_address`, which has one parameter, "Putnum".

The values that `read_address` returns serve as the contents of a virtual table for the PROC SQL SELECT statement.

```
select * from connection to informix
      (execute procedure read_address ("Putnum"));
```

For example, when you try to execute a stored procedure that returns values from a PROC SQL EXECUTE statement, you receive this error message:

```
execute (execute procedure read_address
      ("Putnum")) by informix;
```

ERROR: Informix EXECUTE Error: Procedure

```
(read_address) returns too many values.
```

## Command Restrictions for the SQL Pass-Through Facility

Informix SQL contains extensions to the ANSI-89 standards. Some of these extensions, such as LOAD FROM and UNLOAD TO, are restricted from use by any applications other than the Informix DB-Access product. Specifying these extensions in the PROC SQL EXECUTE statement generates this error:

```
-201
A syntax error has occurred
```

## Examples

This example connects to Informix by using data source `mysrv1`:

```
proc sql;
  connect to informix
    (user=myusr1 password=mypwd1 server=mysrv1);
```

You can use the DBDATASRC environment variable to specify the default data source.

This next example grants UPDATE and INSERT authority to user `gomez` on the Informix ORDERS table. Because the CONNECT statement is omitted, an implicit connection is made. The connection uses a default value of `informix` as the connection alias and default values for the SERVER= argument.

```
proc sql;
  execute (grant update, insert on ORDERS to gomez) by informix;
quit;
```

This example connects to Informix and drops (removes) the table TempData from the database. The alias Temp5 that is specified in the CONNECT statement is used in the EXECUTE statement's BY clause.

```
proc sql;
  connect to informix as temp5
    (server=mysrv1);
  execute (drop table tempdata) by temp5;
  disconnect from temp5;
quit;
```

This example sends an SQL query, shown with highlighting, to the database for processing. The results from the SQL query serve as a virtual table for the PROC SQL FROM clause. DBCON is a connection alias in this example.

```
proc sql;
connect to informix as dbcon
  (user=myusr1 using=mypwd1
  server=mysrv1);
```

```

select *
  from connection to dbcon
  (select empid, lastname, firstname,
  hiredate, salary
   from employees
   where hiredate>='31JAN88') ;


```

```

disconnect from dbcon;
quit;

```

This next example gives the previous query a name and stores it as the PROC SQL view Samples.Hires88. The CREATE VIEW statement appears in highlighting.

```
libname samples 'SAS-library';
```

```

proc sql;
connect to informix as mycon
  (user=myusr1 using=mypwd1
   server=mysrv1);

```

```

create view samples.hires88 as
select *
  from connection to mycon
  (select empid, lastname, firstname,
  hiredate, salary from employees
   where hiredate>='31JAN88') ;

```

```

disconnect from mycon;
quit;

```

This example connects to Informix and executes the stored procedure testproc. The select \* clause displays the results from the stored procedure.

```

proc sql;
  connect to informix as mydb
    (server=mysrv1);
  select * from connection to mydb
    (execute procedure testproc('123456'));
  disconnect from mydb;
quit;

```

---

## Autopartitioning Scheme for Informix

---

### Overview

Autopartitioning for SAS/ACCESS Interface to Informix is a modulo (MOD) function method. For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page [76](#).

---

## Autopartitioning Restrictions

SAS/ACCESS Interface to Informix places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- INTEGER
- SMALLINT
- BIT
- TINYINT
- You can also use DECIMALS with 0-scale columns as the partitioning column.
- Nullable columns are the least preferable.

---

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in a SAS WHERE clause. For example, the following DATA step cannot use a threaded Read to retrieve the data because all numeric columns in the table are in the WHERE clause:

```
data work.locemp;
set trlib.MYEMPS;
where EMPNUM<=30 and ISTENURE=0 and
      SALARY<=35000 and NUMCLASS>2;
run;
```

---

## Using DBSLICEPARM=

SAS/ACCESS Interface to Informix defaults to three threads when you use autopartitioning. However, do not specify a maximum number of threads in [DBSLICEPARM= LIBNAME option on page 226](#) to use for the threaded Read.

This example shows how to use DBSLICEPARM= with the maximum number of threads set to five:

```
libname x informix user=myusr1 using=mypwd1 server=mysrv1;
proc print data=x.dept(dbsliceparm=(ALL,5));
run;
```

---

## Using DBSLICE=

You can achieve the best possible performance when using threaded Reads by specifying the **DBSLICE=** data set option for Informix in your SAS operation. This example shows how to use it.

```
libname x informix user=myusr1 using=mpwd1 server=mysrv1;
data xottest;
set x.invoice(dbslice=("amtbilled<10000000" "amtbilled>=10000000"));
run;
```

---

## Temporary Table Support for Informix

SAS/ACCESS Interface to Informix supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

---

## Passing SAS Functions to Informix

SAS/ACCESS Interface to Informix passes the following SAS functions to Informix for processing if the DBMS driver or client that you are using supports this function. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|       |         |
|-------|---------|
| ABS   | MAX     |
| ARCOS | MDY     |
| ARSIN | MIN     |
| ATAN  | MINUTE  |
| ATAN2 | MONTH   |
| AVG   | SECOND  |
| COS   | SIN     |
| COUNT | SQRT    |
| DATE  | STRIP   |
| DAY   | SUM     |
| EXP   | TAN     |
| HOUR  | TODAY   |
| INT   | WEEKDAY |
| LOG   | YEAR    |
| LOG10 |         |

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to Informix. Due to incompatibility in date and time functions between Informix and SAS, the Informix server might not process them correctly. Check your results to determine whether these functions are working as expected.

DATEPART    TIMEPART

## Passing Joins to Informix

For a multiple libref join to pass to Informix, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (USING=)
- server (SERVER=)

Due to an Informix database limitation, the maximum number of tables that you can specify to perform a join is 22. An error message appears if you specify more than 22 tables.

Informix supports nonstandard outer-join syntax. Outer joins between two tables can be passed to the DBMS, with these restrictions:

- Full outer joins are not supported.
- Only a comparison operator is allowed in an ON clause.
- Outer joins cannot contain a WHERE clause.

## Locking in the Informix Interface

In most cases, SAS spooling is on by default for the Informix interface and provides the data consistency that you need.

To control how the Informix interface handles locking, you can use the **READ\_ISOLATION\_LEVEL= LIBNAME** option. Here are the valid values.

### COMMITTED\_READ

retrieves only committed rows. No locks are acquired, and rows can be locked exclusively for update by other users or processes. This is the default value.

### REPEATABLE\_READ

gives you a shared lock on every row that is selected during the transaction. Other users or processes can also acquire a shared lock, but no other process can modify any row that is selected by your transaction. If you repeat the query during the transaction, you re-read the same information. The shared locks are

released only when the transaction commits or rolls back. Another process cannot update or delete a row that is accessed by using a repeatable read.

#### **DIRTY\_READ**

retrieves committed and uncommitted rows that might include phantom rows. *Phantom rows* are rows that are created or modified by another user or process that might subsequently be rolled back. This type of read is most appropriate for tables that are not frequently updated.

#### **CURSOR\_STABILITY**

gives you a shared lock on the selected row. Another user or process can acquire a shared lock on the same row, but no process can acquire an exclusive lock to modify data in the row. When you retrieve another row or close the cursor, the shared lock is released.

If you specify `READ_ISOLATION_LEVEL= REPEATABLE_READ` or `CURSOR_STABILITY`, it is recommended that you assign a separate libref and clear that libref when you finish working with the tables. This technique minimizes the negative performance impact on other users that occurs when you lock the tables. To clear the libref, include this code:

```
libname libref clear;
```

For current Informix releases, `READ_ISOLATION_LEVEL=` is valid only when transaction logging is enabled. If transaction logging is not enabled, an error is generated when you use this option. Also, locks placed when `READ_ISOLATION_LEVEL= REPEATABLE READ` or `CURSOR_STABILITY` are *not freed* until the libref is cleared.

To see the SQL locking statements that SAS issues to the Informix server, include in your code the `SASTRACE=` system option.

```
option sastrace=',,,d';
```

For more details about Informix locking, see your Informix documentation.

## Naming Conventions for Informix

Most SAS names can be up to 32 characters long. The Informix interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

The `PRESERVE_TAB_NAMES=` and `PRESERVE_COL_NAMES=` options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Informix is not case sensitive, so all names default to lowercase.

Informix objects include tables and columns. They follow these naming conventions.

- Although table and column names must be from 1 to 32 characters, the limitation on some Informix servers might be lower.
- Table and column names must begin with a letter or an underscore (\_) that is followed by letters, numbers, or underscores. Special characters are not supported. However, if you enclose a name in quotation marks and PRESERVE\_TAB\_NAMES=YES (when applicable), it can begin with any character.

Because several problems were found in the Informix ODBC driver that result from using uppercase or mixed case, Informix encourages users to use lowercase for table and column names. Informix currently has no schedule for fixing these known problems.

---

# Data Types for Informix

---

## Overview

Every column in a table has a name and a data type. The data type tells Informix how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Informix data types, null values, and data conversions.

---

## Supported Informix Data Types

---

### Data Types

Here are the data types that the Informix engine supports.

- Character data:

|                       |                        |
|-----------------------|------------------------|
| CHAR( <i>n</i> )      | NVARCHAR( <i>m,n</i> ) |
| NCHAR( <i>n</i> )     | TEXT                   |
| VARCHAR( <i>m,n</i> ) | BYTE                   |

- Numeric data:

|                  |            |
|------------------|------------|
| DECIMAL          | REAL       |
| MONEY            | SMALLFLOAT |
| NUMERIC          | SERIAL     |
| FLOAT            | SMALLINT   |
| DOUBLE PRECISION | INT8       |
| INTEGER          | SERIAL8    |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

**TIP** When the length value of INT8 or SERIAL8 is greater than 15, the last few digits currently do not display correctly due to a display limitation.

- Date, time, and interval data:

DATE            INTERVAL  
DATETIME

## Informix Null Values

Informix has a special value that is called NULL. An Informix NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads an Informix NULL value, it interprets it as a SAS missing value.

If you do not indicate a default value for an Informix column, the default value is NULL. You can specify the keywords NOT NULL after the data type of the column when you create an Informix table to prevent NULL values from being stored in the column. When you create an Informix table with SAS/ACCESS, you can use the **DBNULL=** data set option to indicate whether NULL is a valid value for specified columns.

For more information about how SAS handles NULL values, see [Potential Result Set Differences When Processing Null Data](#) on page 43.

To control how the DBMS handles SAS missing character values, use the **NULLCR=** and **NULLCRVAL=** data set options.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Informix assigns to SAS variables when using the LIBNAME statement to read from an Informix table. These default formats are based on Informix column attributes. To override these default data types, use the **DBTYPE=** data set option on a specific data set.

**Table 25.3** LIBNAME Statement: Default SAS Formats for Informix Data Types

| Informix Column Type                | Default SAS Format                  |
|-------------------------------------|-------------------------------------|
| CHAR( <i>n</i> )                    | \$ <i>n</i>                         |
| DATE                                | DATE9.                              |
| DATETIME <sup>3</sup>               | DATETIME24.5                        |
| DECIMAL                             | <i>m+2.n</i>                        |
| DOUBLE PRECISION                    | none                                |
| FLOAT                               | none                                |
| INTEGER                             | none                                |
| INT8 <sup>2</sup>                   | none                                |
| INTERVAL                            | \$ <i>n</i>                         |
| MONEY                               | none                                |
| NCHAR( <i>n</i> )                   | \$ <i>n</i><br>NLS support required |
| NUMERIC                             | none                                |
| NVARCHAR( <i>m,n</i> ) <sup>1</sup> | \$ <i>m</i><br>NLS support required |
| REAL                                | none                                |
| SERIAL                              | none                                |
| SERIAL8 <sup>2</sup>                | none                                |
| SMALLFLOAT                          | none                                |
| SMALLINT                            | none                                |
| TEXT <sup>1</sup>                   | \$ <i>n</i>                         |
| VARCHAR( <i>m,n</i> ) <sup>1</sup>  | \$ <i>m</i>                         |

<sup>1</sup> Supported only by Informix online databases.<sup>2</sup> The precision of an INT8 or SERIAL8 is 15 digits.<sup>3</sup> If the Informix field qualifier specifies either HOUR, MINUTE, SECOND, or FRACTION as the largest unit, the value is converted to a SAS TIME value. All other values (such as YEAR, MONTH, or DAY) are converted to a SAS DATETIME value.

The following table shows the default Informix data types that SAS/ACCESS applies to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 25.4 LIBNAME Statement: Default Informix Data Types for SAS Variable Formats**

| SAS Variable Format              | Informix Data Type           |
|----------------------------------|------------------------------|
| \$w.                             | CHAR(w).                     |
| w. with SAS format name of NULL  | DOUBLE                       |
| w.d with SAS format name of NULL | DOUBLE                       |
| all other numerics               | DOUBLE                       |
| datetimew.d                      | DATETIME YEAR TO FRACTION(5) |
| datew.                           | DATE                         |
| time.                            | DATETIME HOUR TO SECOND      |

---

## SQL Pass-Through Facility Data Conversions

The [SQL pass-through facility](#) uses the same default conversion formats as the LIBNAME statement. For conversion tables, see “[LIBNAME Statement Data Conversions](#)” on page 986.

---

## Informix Servers

---

### Overview: Informix Database Servers

There are two types of Informix database servers, the Informix OnLine and Informix SE servers. Informix OnLine database servers can support many users and provide tools that ensure high availability, high reliability, and that support critical applications. Informix SE database servers are designed to manage relatively small databases that individuals use privately or that a small number of users share.

---

## Using the DBDATASRC Environment Variable

The SQL pass-through facility supports the DBDATASRC environment variable, which is an extension to the Informix environment variable. If you specify DBDATASRC, you can omit the CONNECT statement. The value of DBDATASRC is used instead of the SERVER= argument in the CONNECT statement. The syntax for specifying DBDATASRC is like the syntax of the SERVER= argument:

Bourne shell:

```
export DBDATABASE='mysrv1'
```

C shell:

```
setenv DBDATASRC mysrv1
```

If you specify DBDATASRC, you can issue a PROC SQL SELECT or EXECUTE statement without first connecting to Informix with the CONNECT statement.

If you omit the CONNECT statement, an implicit connection is performed when the SELECT or EXECUTE statement is passed to Informix.

If you create an SQL view without an explicit CONNECT statement, the view can dynamically connect to different databases, depending on the value of the DBDATASRC environment variable.

---

## Using Fully Qualified Table Names

Informix supports a connection to only one database. If you have data that spans multiple databases, you must use fully qualified table names to work within the Informix single-connection constraints.

In this example, the tables Tab1 and Tab2 reside in different databases, MyDB1 and MyDB2, respectively.

```
proc sql;
    connect to informix
    (server=mysrv1);

    create view tab1v as
        select * from connection
        to informix
        (select * from mydb1.tab1);

    create view tab2v as
        select * from connection
        to informix
        (select * from mydb2.tab2);
quit;

data getboth;
    merge tab1v tab2v;
    by common;
run;
```

Because the tables reside in separate databases, you cannot connect to each database with a PROC SQL CONNECT statement and then retrieve the data in a single step. Using the fully qualified table name (that is, *database.table*) enables you to use any Informix database in the CONNECT statement and access Informix tables in the *same or different* databases in a single SAS procedure or DATA step.

# SAS/ACCESS Interface to JDBC

---

|                                                                   |             |
|-------------------------------------------------------------------|-------------|
| <i>System Requirements for SAS/ACCESS Interface to JDBC</i> ..... | <b>992</b>  |
| <i>Introduction to SAS/ACCESS Interface to JDBC</i> .....         | <b>992</b>  |
| Overview .....                                                    | 992         |
| JDBC Concepts .....                                               | 992         |
| <i>LIBNAME Statement for the JDBC Engine</i> .....                | <b>994</b>  |
| Overview .....                                                    | 994         |
| Arguments .....                                                   | 995         |
| JDBC LIBNAME Statement Examples .....                             | 998         |
| <i>Data Set Options for JDBC</i> .....                            | <b>998</b>  |
| <i>SQL Pass-Through Facility Specifics for JDBC</i> .....         | <b>999</b>  |
| Key Information .....                                             | 999         |
| CONNECT Statement Examples .....                                  | 1000        |
| <i>Temporary Table Support for JDBC</i> .....                     | <b>1001</b> |
| <i>Passing SAS Functions to JDBC</i> .....                        | <b>1001</b> |
| <i>Passing Joins to JDBC</i> .....                                | <b>1002</b> |
| <i>Bulk Loading for JDBC</i> .....                                | <b>1002</b> |
| <i>Naming Conventions for JDBC</i> .....                          | <b>1003</b> |
| <i>Data Types for JDBC</i> .....                                  | <b>1004</b> |
| Overview .....                                                    | 1004        |
| JDBC Null Values .....                                            | 1004        |
| LIBNAME Statement Data Conversions .....                          | 1004        |

---

# System Requirements for SAS/ACCESS Interface to JDBC

You can find information about system requirements for SAS/ACCESS Interface to JDBC in the following locations:

- [System Requirements for SAS/ACCESS Interface to JDBC with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to JDBC

---

## Overview

For available SAS/ACCESS features, see [JDBC supported features on page 106](#). For more information about JDBC, see your JDBC documentation.

SAS/ACCESS Interface to JDBC includes SAS Data Connector to JDBC. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“JDBC Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

---

## JDBC Concepts

---

### Overview

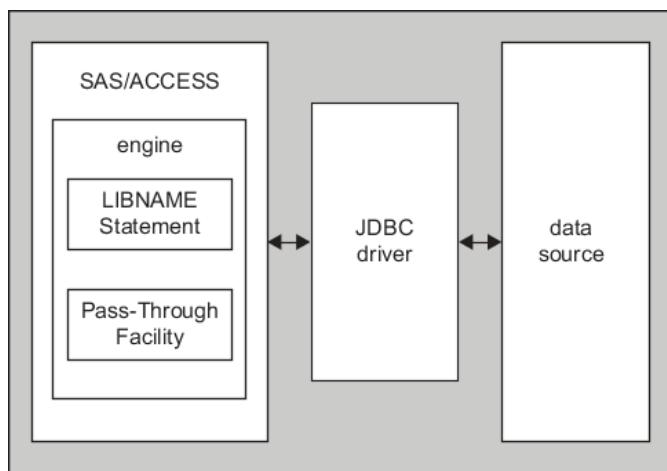
Java Database Connectivity (JDBC) standards provide a common interface to a variety of data sources, including PostgreSQL, MySQL, and other data sources that are compliant with JDBC. The goal of JDBC is to enable access to data from any

application, regardless of which DBMS handles the data. JDBC accomplishes this by inserting a middle layer—which includes a JDBC driver—between an application and the target DBMS. The purpose of this layer is to translate application data queries into commands that the DBMS understands. Specifically, JDBC standards specify application programming interfaces (APIs) that enable applications such as SAS software to access a database. For all of this to work, both the application and the DBMS must be compliant with JDBC. This means that the application must be able to issue JDBC commands, and the DBMS must be able to respond to these.

Here are the basic components and features of JDBC.

The components provide JDBC functionality: the client interface and the JDBC driver for the data source that you want to access, as shown below.

**Figure 26.1** The JDBC Interface to SAS



## JDBC Details

JDBC uses SQL syntax for queries and statement execution, or for statements that are executed as commands. However, all databases that support JDBC are not necessarily SQL databases. For example, many databases do not have system tables. Also, the term *table* can describe a variety of items—including a file, a part of a file, a group of files, a typical SQL table, generated data, or any potential source of data. This is an important distinction. All JDBC data sources respond to a base set of SQL statements such as SELECT, INSERT, UPDATE, DELETE, CREATE, and DROP in their simplest forms. However, some databases do not support other statements and more complex forms of SQL statements.

To maximize your chances of success, your JDBC driver must support the call sequences that SAS/ACCESS Interface to JDBC sends to the driver. Specifically, your JDBC driver must support these JDBC classes:

- java.sql.Connection
- java.sql.Driver
- java.sql.DriverManager (getConnection method)
- java.sql.PreparedStatement
- java.sql.ResultSet
- java.sql.ResultSetMetaData

```
java.sql.SQLException
java.sql.Statement
```

SAS/ACCESS Interface to JDBC sends a sequence of JDBC calls to the JDBC driver that you have chosen. The types and sequence in which these calls are made are compliant with the JDBC specification. If your JDBC driver fails to return the correct result or fails to work with SAS/ACCESS Interface to JDBC, try these suggestions:

- Be sure to run the current versions of your JDBC client components.
- Try to connect using a query tool that is not SAS. Most third-party JDBC driver sources include such a query tool with their offerings. However, keep in mind that in some cases you might be able to connect using a query tool (that is not SAS) but not with SAS/ACCESS Interface to JDBC. This is because SAS calls might make a wider range of JDBC calls than a JDBC query tool other than SAS would make.
- Contact SAS Technical Support. SAS Technical Support offers additional tools that can help you identify the root of the problems that are related to your JDBC client and subsequently debug them.
- Attempt to use certain SAS/ACCESS Interface to JDBC options or alternative engines to SAS/ACCESS Interface to JDBC.
- If you have determined that your JDBC client issues are not related to SAS, report your debugging results to your JDBC client providers. If you received your JDBC client components from a commercial JDBC driver vendor, you can work through that vendor's technical support. If you use freeware or open-source JDBC client components—where formal technical support is not always available—your only recourse might be to communicate with the freeware user community.

SAS has not validated all JDBC drivers on the market and therefore makes no claims of certification or support.

## LIBNAME Statement for the JDBC Engine

### Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to JDBC supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing JDBC.

**LIBNAME** *libref* **JDBC** <connection-options> <LIBNAME-options>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in *SAS Global Statements: Reference*.

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### **JDBC**

specifies the SAS/ACCESS engine name for the JDBC interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS.

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

#### **USER=<'>JDBC-user-name<'>**

lets you connect to a JDBC database with a user ID that is different from the default ID. USER= is optional.

---

**Note:** USER= must be specified as a connection option and cannot be specified via the URL= option.

---

#### Alias: UID=

#### **PASSWORD=<'>JDBC-password<'>**

specifies the JDBC password that is associated with your user ID. PASSWORD= is optional. If the password contains spaces or nonalphanumeric characters, enclose it in quotation marks. If you do not want to enter your JDBC password in uncoded text on this statement, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

---

**Note:** PASSWORD= must be specified as a connection option and cannot be specified via the URL= option.

---

#### Alias: PWD=

#### **DRIVERCLASS="value"**

specifies the JDBC driver to use for your database connection.

---

**Note:** Support for this option was added in SAS Viya 3.4.

---

#### Alias: CLASS=

**Example:** `driverclass="org.postgresql.Driver"`

**CLASSPATH="JAR-path"**

specifies the path to the JDBC JAR files. The value that you specify for CLASSPATH= overrides the value that is set for the SAS\_ACCESS\_CLASSPATH environment variable.

---

**Note:** Support for this option was added in SAS Viya 3.4.

---

**Default:** SAS\_ACCESS\_CLASSPATH environment variable value

**Requirement:** You must set a value for the CLASSPATH= connection option or the SAS\_ACCESS\_CLASSPATH environment variable.

**Example:** options set=CLASSPATH='JAR-path';

**URL="jdbc:driver-name://driver-connection-options"**

specifies the JDBC connection string to use for your database connection.

**Example:** url="jdbc:postgresql://<server>:<port>/test"

See your JDBC driver documentation for a list of the JDBC connection options that your JDBC driver supports.

**LIBNAME-options**

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to JDBC, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 26.1** SAS/ACCESS LIBNAME Options for JDBC

| Option               | Default Value | Valid in CONNECT Statement |
|----------------------|---------------|----------------------------|
| ACCESS=              | none          |                            |
| AUTHDOMAIN=          | none          |                            |
| BULKLOAD=            | YES           | •                          |
| CONNECTION=          | SHAREDREAD    | •                          |
| DBCONINIT=           | none          | •                          |
| DBCONTERM=           | none          | •                          |
| DBCREATE_TABLE_OPTS= | none          |                            |
| DBGEN_NAME=          | DBMS          | •                          |
| DBLIBINIT=           | none          |                            |
| DBLIBTERM=           | none          |                            |

| Option              | Default Value                                                   | Valid in<br>CONNECT<br>Statement |
|---------------------|-----------------------------------------------------------------|----------------------------------|
| DBMAX_TEXT=         | 1024                                                            | •                                |
| DBMSTEMP=           | NO                                                              |                                  |
| DBSASLABEL=         | COMPAT                                                          |                                  |
| DEFER=              | NO                                                              | •                                |
| DIRECT_SQL=         | YES                                                             |                                  |
| INSERTBUFF=         | based on row length                                             |                                  |
| LOGIN_TIMEOUT=      | 0                                                               | •                                |
| MULTI_DATASRC_OPT=  | NONE                                                            |                                  |
| POST_STMT_OPTS=     | none                                                            |                                  |
| PRESERVE_COL_NAMES= | NO<br><br>(see “ <a href="#">Naming Conventions for JDBC</a> ”) |                                  |
| PRESERVE_TAB_NAMES= | NO<br><br>see (“ <a href="#">Naming Conventions for JDBC</a> ”) |                                  |
| QUERY_TIMEOUT=      | 0                                                               | •                                |
| QUOTE_CHAR=         | none                                                            |                                  |
| READBUFF=           | automatically calculated<br>based on row length                 | •                                |
| SCHEMA=             | none                                                            |                                  |
| SPOOL=              | YES                                                             |                                  |
| SQL_FUNCTIONS=      | none                                                            |                                  |
| SQL_FUNCTIONS_COPY= | none                                                            |                                  |
| SQLGENERATION=      | none                                                            |                                  |
| SUB_CHAR=           | none                                                            |                                  |
| TRANSCODE_FAIL=     | ERROR                                                           | •                                |

## JDBC LIBNAME Statement Examples

In this example, USER=, PASSWORD=, URL=, and DRIVERCLASS= are connection options.

```
libname mydblib JDBC user=myusr1 password=mypwd1
      url="jdbc:postgresql://<server>:<port>/myDB"
      driverclass="org.postgresql.Driver";
```

In this next example, the libref MYLIB uses the JDBC engine to connect to a PostgreSQL database. The connection options are DRIVERCLASS=, URL=, USER=, PASSWORD=, and CLASSPATH=. The CLASSPATH= LIBNAME option indicates that the vendor's JDBC driver JAR file is contained in the c:\lib directory.

```
libname x JDBC driverclass="org.postgresql.Driver"
      URL="jdbc:postgresql://myserver:myport/<postgres-database>"
      user=myuser
      password="mypwd1" classpath="c:\lib";

proc print data=x.customers;
  where state='CA';
run;
```

## Data Set Options for JDBC

All SAS/ACCESS data set options in this table are supported for JDBC. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

*Table 26.2 SAS/ACCESS Data Set Options for JDBC*

| Option               | Default Value        |
|----------------------|----------------------|
| BULKLOAD=            | LIBNAME option value |
| DBCONDITION=         | none                 |
| DBCREATE_TABLE_OPTS= | LIBNAME option value |
| DBFORCE=             | NO                   |
| DBGEN_NAME=          | DBMS                 |
| DBLABEL=             | NO                   |
| DBLARGETABLE=        | none                 |

| Option              | Default Value             |
|---------------------|---------------------------|
| DBMAX_TEXT=         | 1024                      |
| DBNULLWHERE=        | LIBNAME option value      |
| DBSASLABEL=         | COMPAT                    |
| DBSASTYPE=          | see “Data Types for JDBC” |
| DBTYPE=             | see “Data Types for JDBC” |
| INSERTBUFF=         | LIBNAME option value      |
| POST_STMT_OPTS=     | none                      |
| POST_TABLE_OPTS=    | none                      |
| PRE_STMT_OPTS=      | none                      |
| PRE_TABLE_OPTS=     | none                      |
| PRESERVE_COL_NAMES= | LIBNAME option value      |
| QUERY_TIMEOUT=      | LIBNAME option value      |
| READBUFF=           | LIBNAME option value      |
| SCHEMA=             | LIBNAME option value      |
| SUB_CHAR=           | none                      |
| TRANSCODE_FAIL=     | LIBNAME option value      |

---

## SQL Pass-Through Facility Specifics for JDBC

---

### Key Information

For general information about this feature, see “SQL Pass-Through Facility” on page 690.

Here are the SQL pass-through facility specifics for the JDBC interface.

- The *dbms-name* is JDBC.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to JDBC. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default JDBC alias is used. The functionality of multiple connections to the same JDBC data source might be limited by the particular data source driver.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection-options](#).
- On some DBMSs, the *DBMS-SQL-query* argument can be a DBMS-specific SQL EXECUTE statement that executes a DBMS stored procedure. However, if the stored procedure contains more than one query, only the first query is processed.

## CONNECT Statement Examples

These examples use JDBC to connect to a PostgreSQL data source. The first example uses the connection method that is guaranteed to be present at the lowest level of JDBC conformance.

```
proc sql;
  connect to jdbc as C (driverclass="org.postgresql.Driver"
    URL="jdbc:postgresql://<PostgreSQL-server>:<port>/<PostgreSQL-
    database>"'
    user=user1 password="myPwd" classpath="c:\lib");
  select * from connection to c (select * from class);
  quit;
```

If a LIBNAME statement has been specified, you can use the libref to connect to the JDBC data source.

```
libname x jdbc driverclass="org.postgresql.Driver"
  URL="jdbc:postgresql://<PostgreSQL-server>:<port>/<PostgreSQL-
  database>"'
  user=user1 password="myPwd" classpath="c:\lib";

proc sql;
  connect using x;
  select * from connection to x (select * from class);
  quit;

proc sql;
  connect using x;
  execute (create table test12(c char(10))) by x;
  quit;
```

---

# Temporary Table Support for JDBC

SAS/ACCESS Interface to JDBC supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

---

## Passing SAS Functions to JDBC

SAS/ACCESS Interface to JDBC passes down database-specific functions that take scalar values as arguments to your data source. The list of functions that SAS passes down is a subset of the X/Open Group CLI specification. If a data source supports a scalar function, its driver is also expected to support this function using JDBC escape syntax. That is, the specified function name is passed down in the format {fn <function-name>}.

Where the JDBC function name differs from the SAS SQL function name, the JDBC name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                                      |                 |
|--------------------------------------|-----------------|
| ** (POWER)                           | LOG             |
| ABS                                  | LOG10           |
| ACOS                                 | LOWCASE (LCASE) |
| ASIN                                 | MAX             |
| ATAN                                 | MIN             |
| ATAN2                                | MINUTE          |
| AVG                                  | MONTH           |
| CAT (CONCAT)                         | PI              |
| CEIL (CEILING)                       | QTR (QUARTER)   |
| CHAR                                 | RADIANS         |
| COS                                  | RAND            |
| COT                                  | SECOND          |
| COUNT                                | SIGN            |
| DATE (CURDATE)                       | SIN             |
| DATETIME (CURTIME or NOW – see Note) | SOUNDEX         |
| DTEXTDAY (DAYNAME)                   | SQRT            |
| DTEXTMONTH (MONTHNAME)               | SUBSTR          |
| DTEXTWEEKDAY (DAYOFWEEK)             | SUM             |
| EXP                                  | TAN             |
| FLOOR                                | TRIMN (RTRIM)   |
| HOUR                                 | UPCASE (UCASE)  |
| INDEX (LOCATE)                       | WEEK            |
| LEFT (LTRIM)                         | YEAR            |

## LENGTH

---

**Note:** The DATETIME function passes either the CURTIME or NOW function, depending on the database that you are connecting to using JDBC. You can specify SASTRACE=',,,d' and MSGLEVEL=i in the OPTIONS statement to see which function is passed to your database. For more information, see “[Tracing and Evaluating SQL Generation](#)” on page 56.

---

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to JDBC. Due to incompatibility in date and time functions between JDBC and SAS, JDBC might not process them correctly. Check your results to determine whether these functions are working as expected.

MOD  
REPEAT  
ROUND

---

## Passing Joins to JDBC

For a multiple libref join to pass to JDBC, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- URL (URL=)
- driver class (DRIVERCLASS=)

Outer joins between two or more tables can be passed to the DBMS. However, the outer joins must not be mixed with inner joins in a query.

PROC SQL attempts to pass joins across databases to the data source. However, the data source might or might not support this query. If the query cannot be completed in the data source, then the query is processed in SAS.

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

## Bulk Loading for JDBC

SAS/ACCESS Interface to JDBC implements the **BULKLOAD= LIBNAME** option differently from other SAS/ACCESS interfaces. For JDBC, an external bulk-loading program is not called to implement bulk loading. Instead, **BULKLOAD=YES** means that data is loaded with the JDBC batch update facility. Batch update is generally the

fastest method to insert data into a database using a JDBC driver when there is not an external program for bulk loading. BULKLOAD=YES is the default for SAS/ACCESS Interface to JDBC.

Some JDBC drivers do not support batch update. If the JDBC driver reports via the JDBC DataBaseMetaData that batch updates are not supported, then SAS/ACCESS Interface to JDBC automatically acts as if BULKLOAD=NO. Alternatively, you can manually specify BULKLOAD=NO. When BULKLOAD=NO, SAS/ACCESS Interface to JDBC attempts to insert data into the target database using an SQL query that uses the `INSERT INTO <table> VALUES <values>` clause. Some databases support multiple `VALUES` clauses that so that you can insert many rows in a single `INSERT` clause.

## Naming Conventions for JDBC

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Because JDBC is an application programming interface (API) rather than a database, table names and column names are determined at run time. Most SAS names can be up to 32 characters long. SAS/ACCESS Interface to JDBC supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, SAS truncates them to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The `PRESERVE_TAB_NAMES=` and `PRESERVE_COL_NAMES=` options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).)

This example specifies PostgreSQL as the DBMS.

```
libname x jdbc driverclass="org.postgresql.Driver"
      URL="jdbc:postgresql://<myserver>:<myport>/<postgres-database>" user=myuser
      password="mypwd1" classpath="c:\lib" preserve_col_names=no;

data x.a;
  xAxis=1;
  yAxis=2;
run;
```

PostgreSQL stores identifiers in lowercase unless the identifier is quoted. This example would therefore produce a PostgreSQL table named `a` with columns named `xaxis` and `yaxis`.

**Note:** The default value for `PRESERVE_COL_NAMES=` for JDBC is NO, so specifying it in the example above is not required.

If PRESERVE\_COL\_NAMES=YES, then PROC SQL puts identifiers in quotation marks and names are preserved. When you use quotation marks, PostgreSQL becomes case-sensitive.

## Data Types for JDBC

### Overview

Every column in a table has a name and a data type. The data type tells the DBMS how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about JDBC null and default values and data conversions.

### JDBC Null Values

Many relational database management systems have a special value called NULL. A DBMS NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a DBMS NULL value, it interprets it as a SAS missing value.

In most relational databases, columns can be specified as NOT NULL so that they require data (they cannot contain NULL values). When a column is specified as NOT NULL, the DBMS does not add a row to the table unless the row has a value for that column.

JDBC mirrors the behavior of the underlying DBMS with regard to NULL values. See the documentation for your DBMS for information about how it handles NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

### LIBNAME Statement Data Conversions

This table shows all data types and default SAS formats that SAS/ACCESS Interface to JDBC supports.

**Table 26.3 JDBC Data Types and Default SAS Formats**

| JDBC Data Type | Default SAS Format |
|----------------|--------------------|
| CHAR           | \$w.               |

| JDBC Data Type | Default SAS Format                                                 |
|----------------|--------------------------------------------------------------------|
| VARCHAR        | \$w.                                                               |
| LONGVARCHAR    | \$w.                                                               |
| BLOB           | \$w.                                                               |
| CLOB           | \$w.                                                               |
| BINARY         | \$w. <sup>1</sup>                                                  |
| VARBINARY      | \$w. <sup>1</sup>                                                  |
| LONGVARBINARY  | \$w. <sup>1</sup>                                                  |
| DECIMAL        | w. or w.d or none if w and d are not specified                     |
| NUMERIC        | w. or w.d or none if w and d are not specified                     |
| INTEGER        | 11.                                                                |
| BIGINT         | 20.                                                                |
| SMALLINT       | 6.                                                                 |
| TINYINT        | 4.                                                                 |
| BIT            | 1.                                                                 |
| REAL           | none                                                               |
| FLOAT          | none                                                               |
| DOUBLE         | none                                                               |
| DATE           | DATE9.                                                             |
| TIME           | TIME8.<br>JDBC cannot support fractions of seconds for time values |
| TIMESTAMP      | DATETIMEw.d where w and d depend on precision                      |
| ARRAY          | none                                                               |
| DISTINCT       | Character values are mapped to \$w.                                |

| JDBC Data Type | Default SAS Format                      |
|----------------|-----------------------------------------|
|                | Numeric values are mapped to <i>w</i> . |
| JAVA_OBJECT    | none                                    |
| REF            | none                                    |
| STRUCT         | none                                    |

**1** Because the JDBC driver does the conversion, this field is displayed as if the \$HEX*w*. format were applied.

This table shows the default data types that SAS/ACCESS Interface to JDBC uses when creating tables. SAS/ACCESS Interface to JDBC lets you specify non-default data types by using the **DBTYPE=** data set option.

**Table 26.4 Default JDBC Output Data Types**

| SAS Variable Format | Default JDBC Data Type                                   |
|---------------------|----------------------------------------------------------|
| <i>w.d</i>          | DOUBLE or NUMERIC using <i>w.d</i> if the DBMS allows it |
| \$ <i>w</i> .       | VARCHAR using <i>w</i>                                   |
| datetime formats    | TIMESTAMP                                                |
| date formats        | DATE                                                     |
| time formats        | TIME                                                     |

# SAS/ACCESS Interface to Microsoft SQL Server

|                                                                                       |      |
|---------------------------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to Microsoft SQL Server</i> . . . . . | 1008 |
| <i>Introduction to SAS/ACCESS Interface to Microsoft SQL Server</i> . . . . .         | 1008 |
| <i>LIBNAME Statement for the Microsoft SQL Server Engine</i> . . . . .                | 1009 |
| Overview . . . . .                                                                    | 1009 |
| Arguments . . . . .                                                                   | 1009 |
| Microsoft SQL Server LIBNAME Statement Examples . . . . .                             | 1014 |
| <i>Data Set Options for Microsoft SQL Server</i> . . . . .                            | 1015 |
| <i>SQL Pass-Through Facility Specifics for Microsoft SQL Server</i> . . . . .         | 1018 |
| Key Information . . . . .                                                             | 1018 |
| CONNECT Statement Examples . . . . .                                                  | 1018 |
| Connection to Component Examples . . . . .                                            | 1019 |
| <i>Passing Joins to Microsoft SQL Server</i> . . . . .                                | 1019 |
| <i>Passing SAS Functions to Microsoft SQL Server</i> . . . . .                        | 1020 |
| <i>DBLOAD Procedure Specifics for Microsoft SQL Server</i> . . . . .                  | 1021 |
| Overview . . . . .                                                                    | 1021 |
| Examples . . . . .                                                                    | 1022 |
| <i>Temporary Table Support for Microsoft SQL Server</i> . . . . .                     | 1022 |
| <i>Locking in the Microsoft SQL Server Interface</i> . . . . .                        | 1023 |
| <i>Bulk Loading with Microsoft SQL Server</i> . . . . .                               | 1024 |
| Overview of Bulk Loading Setup . . . . .                                              | 1024 |
| Bulk Loading to Most Microsoft SQL Server Databases . . . . .                         | 1024 |
| Bulk Loading to Azure Synapse Analytics . . . . .                                     | 1025 |
| Authentication for Bulk Loading to Azure . . . . .                                    | 1026 |
| <i>Naming Conventions for Microsoft SQL Server</i> . . . . .                          | 1026 |
| <i>Data Types for Microsoft SQL Server</i> . . . . .                                  | 1027 |
| Overview . . . . .                                                                    | 1027 |
| Microsoft SQL Server Null Values . . . . .                                            | 1027 |
| LIBNAME Statement Data Conversions . . . . .                                          | 1027 |

|                                                       |             |
|-------------------------------------------------------|-------------|
| Conversion to Money Data Types .....                  | 1030        |
| <i>Sample Programs for Microsoft SQL Server</i> ..... | <b>1030</b> |

---

# System Requirements for SAS/ACCESS Interface to Microsoft SQL Server

You can find information about system requirements for SAS/ACCESS Interface to Microsoft SQL Server in the following locations:

- [System Requirements for SAS/ACCESS Interface to Microsoft SQL Server with SAS 9.4](#)
  - [SAS Viya System Requirements](#)
  - [Third-Party Software Requirements](#)
- 

# Introduction to SAS/ACCESS Interface to Microsoft SQL Server

For available SAS/ACCESS features, see [Microsoft SQL Server supported features](#). For more information about Microsoft SQL Server, see your Microsoft SQL Server documentation.

In [SAS 9.4M4](#), support that enables you to connect to row-based Microsoft Azure SQL database tables was added. As a best practice, set the EnableScrollableCursors ODBC driver option to 3.

SAS/ACCESS Interface to Microsoft SQL Server has been tested and certified against Data Direct Technologies Connect ODBC and Data Direct Sequelink ODBC products.

SAS/ACCESS Interface to Microsoft SQL Server includes SAS Data Connector to Microsoft SQL Server. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“Microsoft SQL Server Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

# LIBNAME Statement for the Microsoft SQL Server Engine

## Overview

This section describes the LIBNAME statement as supported in SAS/ACCESS Interface to Microsoft SQL Server. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Microsoft SQL Server.

**LIBNAME** *libref* **sqlsvr** <connection-options> <LIBNAME-options>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### **sqlsvr**

specifies the SAS/ACCESS engine name for the Microsoft SQL Server interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to Microsoft SQL Server in many different ways. Specify only one of these methods for each connection because they are mutually exclusive.

- USER=, PASSWORD=, and DATASRC=
- COMPLETE=
- NOPROMPT=
- PROMPT=
- REQUIRED=

Here is how these options are defined.

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

**USER=<'>*user-name*<'**

lets you connect to Microsoft SQL Server with a user ID that is different from the default ID.

Alias: **UID=**

Usage: optional

**PASSWORD=<'>*password*<'**

specifies the Microsoft SQL Server password that is associated with your user ID.

Alias: **PWD=**

Usage: optional

**DATASRC=<'>*SQL-Server-data-source*<'**

specifies the Microsoft SQL Server data source to which you want to connect. For UNIX platforms, data sources must be configured by modifying the .ODBC.ini file. This option indicates that the connection is attempted using the ODBC SQLConnect API, which requires a data source name. You can also use a user ID and password with DSN=. This API is guaranteed to be present in all drivers.

Alias: **DATABASE=**, **DB=**, **DSN=**

**COMPLETE=<'>*SQL-Server-connection-options*<'**

specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, you are prompted with a dialog box that displays the values from the COMPLETE= connection string. You can edit any field before you connect to the database. See your driver documentation for more details.

Restriction: This option is not available on UNIX platforms.

**NOPROMPT=<'>*SQL-Server-connection-options*<'**

specifies connection options for your database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned. No dialog box is displayed to help you with the connection string.

Aliases: **CONNECTSTR=**, **CONNECT\_STRING=**

**PROMPT=<'>*SQL-Server-connection-options*<'**

specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately try to connect to the DBMS. Instead, it displays a dialog box in SAS Display Manager that contains the values that you entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the database.

Restriction: This option is not available on UNIX platforms.

**REQUIRED=<'>*SQL-Server-connection-options*<'**

specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, a dialog box prompts you for the connection options. REQUIRED= lets you modify only required fields in the dialog box.

Restriction: This option is not available on UNIX platforms.

*LIBNAME-options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Microsoft SQL Server, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 27.1** SAS/ACCESS LIBNAME Options for Microsoft SQL Server

| Option              | Default Value                                                            | Valid in CONNECT |
|---------------------|--------------------------------------------------------------------------|------------------|
| ACCESS=             | none                                                                     |                  |
| AUTHDOMAIN=         | none                                                                     |                  |
| AUTOCOMMIT=         | NO                                                                       | •                |
| BL_ACCOUNTNAME=     | none                                                                     | •                |
| BL_APPLICATIONID=   | none                                                                     | •                |
| BL_COMPRESS=        | NO                                                                       | •                |
| BL_DEFAULT_DIR=     | temporary file directory that is specified by the UTILLOC= system option | •                |
| BL_DELETE_DATAFILE= | YES                                                                      | •                |
| BL_DELIMITER=       | bell character (ASCII 0x07)                                              | •                |
| BL_DNSSUFFIX=       | dfs.core.windows.net                                                     | •                |
| BL_FILESYSTEM=      | none                                                                     | •                |
| BL_FOLDER=          | none                                                                     | •                |
| BL_IDENTITY=        | none                                                                     | •                |
| BL_LOG=             | none                                                                     | •                |
| BL_MAXERRORS=       | 0                                                                        | •                |
| BL_OPTIONS=         | none                                                                     | •                |
| BL_SECRET=          | none                                                                     | •                |
| BL_TIMEOUT=         | 60                                                                       | •                |

| Option               | Default Value                                                                              | Valid in CONNECT |
|----------------------|--------------------------------------------------------------------------------------------|------------------|
| BL_USE_ESCAPE=       | NO                                                                                         | •                |
| BL_USE_LOG=          | NO                                                                                         | •                |
| BULKLOAD=            | NO                                                                                         | •                |
| CONNECTION=          | UNIQUE when the data source supports only one cursor per connection; otherwise, SHAREDREAD | •                |
| CONNECTION_GROUP=    | none                                                                                       | •                |
| CURSOR_TYPE=         | DYNAMIC                                                                                    | •                |
| DATETIME2=           | NO                                                                                         | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows)                                         |                  |
| DBCONINIT=           | none                                                                                       | •                |
| DBCONTERM=           | none                                                                                       | •                |
| DBCREATE_TABLE_OPTS= | none                                                                                       |                  |
| DBGEN_NAME=          | DBMS                                                                                       | •                |
| DBINDEX=             | NO                                                                                         |                  |
| DBLIBINIT=           | none                                                                                       |                  |
| DBLIBTERM=           | none                                                                                       |                  |
| DBMAX_TEXT=          | 1024                                                                                       | •                |
| DBMSTEMP=            | NO                                                                                         |                  |
| DBNULLKEYS=          | YES                                                                                        |                  |
| DBNULLWHERE=         | YES                                                                                        |                  |
| DBPROMPT=            | NO                                                                                         | •                |
| DBSLICEPARM=         | NONE                                                                                       |                  |
| DEFER=               | NO                                                                                         | •                |

| Option                    | Default Value                                            | Valid in CONNECT |
|---------------------------|----------------------------------------------------------|------------------|
| DELETE_MUTL_ROWS=         | NO                                                       |                  |
| DIRECT_EXE=               | none                                                     |                  |
| DIRECT_SQL=               | YES                                                      |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                       |                  |
| INSERT_SQL=               | YES                                                      |                  |
| INSERTBUFF=               | 1                                                        |                  |
| KEYSET_SIZE=              | 0                                                        |                  |
| LOGIN_TIMEOUT=            | 0                                                        | •                |
| MULTI_DATASRC_OPT=        | NONE                                                     |                  |
| POST_STMT_OPTS=           | none                                                     |                  |
| PRESERVE_COL_NAMES=       | see “Naming Conventions for Microsoft SQL Server”        |                  |
| PRESERVE_COMMENTS=        | NO                                                       | •                |
| PRESERVE_GUID=            | YES                                                      | •                |
| PRESERVE_TAB_NAMES=       | see “Naming Conventions for Microsoft SQL Server”        |                  |
| QUALIFIER=                | none                                                     |                  |
| QUERY_TIMEOUT=            | 0                                                        | •                |
| QUOTE_CHAR=               | none                                                     |                  |
| READBUFF=                 | 0                                                        | •                |
| READ_ISOLATION_LEVEL=     | RC (see “Locking in the Microsoft SQL Server Interface”) | •                |
| READ_LOCK_TYPE=           | ROW                                                      | •                |
| REREAD_EXPOSURE=          | NO                                                       | •                |
| SCHEMA=                   | none                                                     |                  |

| Option                  | Default Value                                            | Valid in CONNECT |
|-------------------------|----------------------------------------------------------|------------------|
| SPOOL=                  | YES                                                      |                  |
| SQL_FUNCTIONS=          | none                                                     |                  |
| SQL_FUNCTIONS_COPY=     | none                                                     |                  |
| SQLGENERATION=          | none                                                     |                  |
| STRINGDATES=            | NO                                                       | •                |
| TRACE=                  | NO                                                       | •                |
| TRACEFILE=              | none                                                     | •                |
| UPDATE_ISOLATION_LEVEL= | RC (see “Locking in the Microsoft SQL Server Interface”) |                  |
| UPDATE_LOCK_TYPE=       | ROW                                                      | •                |
| UPDATE_MULT_ROWS=       | NO                                                       |                  |
| UPDATE_SQL=             | driver-specific                                          |                  |
| USE_ODBC_CL=            | NO                                                       | •                |
| UTILCONN_TRANSIENT=     | NO                                                       |                  |
| WARN_BIGINT=            | NO                                                       | •                |

## Microsoft SQL Server LIBNAME Statement Examples

In this example, USER= and PASSWORD= are connection options.

```
libname mydblib sqldrvr user=myusr1 password=mypwd1 dsn=sqlserver;
```

In this next example, the libref MYDBLIB connects to a Microsoft SQL Server database using the NOPROMPT= option.

```
libname mydblib sqldrvr noprompt="uid=myusr1;
pwd=mypwd1; dsn=sqlservr;" stringdates=yes;

proc print data=mydblib.customers;
  where state='CA';
run;
```

# Data Set Options for Microsoft SQL Server

All SAS/ACCESS data set options in this table are supported for Microsoft SQL Server. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 27.2** SAS/ACCESS Data Set Options for Microsoft SQL Server

| Option              | Default Value        |
|---------------------|----------------------|
| BL_ACCOUNTNAME=     | LIBNAME option value |
| BL_APPLICATIONID=   | LIBNAME option value |
| BL_COMPRESS=        | LIBNAME option value |
| BL_DEFAULT_DIR=     | LIBNAME option value |
| BL_DELETE_DATAFILE= | LIBNAME option value |
| BL_DELIMITER=       | LIBNAME option value |
| BL_DNSSUFFIX=       | LIBNAME option value |
| BL_FILESYSTEM=      | LIBNAME option value |
| BL_FOLDER=          | LIBNAME option value |
| BL_IDENTITY=        | LIBNAME option value |
| BL_LOG=             | LIBNAME option value |
| BL_MAXERRORS=       | LIBNAME option value |
| BL_NUM_DATAFILES=   | 2                    |
| BL_OPTIONS=         | LIBNAME option value |
| BL_SECRET=          | LIBNAME option value |
| BL_TIMEOUT=         | LIBNAME option value |
| BL_USE_ESCAPE=      | LIBNAME option value |

| <b>Option</b>        | <b>Default Value</b>                      |
|----------------------|-------------------------------------------|
| BL_USE_LOG=          | LIBNAME option value                      |
| BULKLOAD=            | LIBNAME option value                      |
| CURSOR_TYPE=         | LIBNAME option value                      |
| DATETIME2=           | NO                                        |
| DBCOMMIT=            | LIBNAME option value                      |
| DBCONDITION=         | none                                      |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                      |
| DBFORCE=             | NO                                        |
| DBGEN_NAME=          | DBMS                                      |
| DBINDEX=             | NO                                        |
| DBKEY=               | none                                      |
| DBLABEL=             | NO                                        |
| DBLARGETABLE=        | none                                      |
| DBMAX_TEXT=          | 1024                                      |
| DBNULL=              | YES                                       |
| DBNULLKEYS=          | LIBNAME option value                      |
| DBNULLWHERE=         | LIBNAME option value                      |
| DBPROMPT=            | LIBNAME option value                      |
| DBSASLABEL=          | COMPAT                                    |
| DBSASTYPE=           | see “Data Types for Microsoft SQL Server” |
| DBSLICE=             | none                                      |
| DBSLICEPARM=         | NONE                                      |
| DBTYPE=              | see “Data Types for Microsoft SQL Server” |

| Option                    | Default Value        |
|---------------------------|----------------------|
| ERRLIMIT=                 | 1                    |
| IGNORE_READ_ONLY_COLUMNS= | NO                   |
| INSERT_SQL=               | LIBNAME option value |
| INSERTBUFF=               | LIBNAME option value |
| KEYSET_SIZE=              | LIBNAME option value |
| NULLCHAR=                 | SAS                  |
| NULLCHARVAL=              | a blank character    |
| POST_STMT_OPTS=           | none                 |
| POST_TABLE_OPTS=          | none                 |
| PRE_STMT_OPTS=            | none                 |
| PRE_TABLE_OPTS=           | none                 |
| PRESERVE_COL_NAMES=       | LIBNAME option value |
| QUALIFIER=                | LIBNAME option value |
| QUERY_TIMEOUT=            | LIBNAME option value |
| READBUFF=                 | LIBNAME option value |
| READ_ISOLATION_LEVEL=     | LIBNAME option value |
| READ_LOCK_TYPE=           | LIBNAME option value |
| SASDATEFMT=               | none                 |
| SCHEMA=                   | LIBNAME option value |
| UPDATE_ISOLATION_LEVEL=   | LIBNAME option value |
| UPDATE_LOCK_TYPE=         | LIBNAME option value |
| UPDATE_SQL=               | LIBNAME option value |

# SQL Pass-Through Facility Specifics for Microsoft SQL Server

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Microsoft SQL Server interface under UNIX hosts.

- The *dbms-name* is `SQLSVR`.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Microsoft SQL Server. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default alias is used. The functionality of multiple connections to the same Microsoft SQL Server data source might be limited by the particular data source driver.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).
- See [Table 27.1 on page 1011](#) to see the LIBNAME options that are automatically passed to the CONNECT statement in PROC SQL.
- The *DBMS-SQL-query* argument can be a DBMS-specific SQL EXECUTE statement that executes a DBMS stored procedure. However, if the stored procedure contains more than one query, only the first query is processed.

## CONNECT Statement Examples

These examples connect to a data source that is configured under the data source name `User's Data` using the alias `USER1`. The first example uses the connection method that is guaranteed to be present at the lowest level of conformance. Note that `DATASRC=` names can contain quotation marks and spaces.

```
proc sql;
  connect to sqldrvr as user1
  (datasrc="User's Data" user=myusr1 password=mypwd1);
```

This example uses the connection method that represents a more advanced level of Microsoft SQL Server ODBC conformance. It uses the input dialog box that is provided by the driver. The `DSN=` and `UID=` arguments are within the connection string. The SQL pass-through facility therefore does not parse them but instead passes them to the ODBC driver manager.

```
proc sql;
  connect to SQLSVR as user1
    (required = "dsn=User's Data; uid=myusr1");
```

In this example, you can select any data source that is configured on your machine. The example uses the connection method that represents a more advanced level of Microsoft SQL Server ODBC conformance, Level 1. When connection succeeds, the connection string is returned in the SQLXMSG and SYSDBMSG macro variables. It can then be stored if you use this method to configure a connection for later use.

```
proc sql;
  connect to SQLSVR (required);
```

This example prompts you to specify the information that is required to make a connection to the DBMS. You are prompted to provide the data source name, user ID, and password in the dialog boxes that are displayed.

```
proc sql;
  connect to SQLSVR (prompt);
quit;
```

## Connection to Component Examples

This example sends Microsoft SQL Server 6.5 (configured under the data source name "SQL Server") an SQL query for processing. The results from the query serve as a virtual table for the PROC SQL FROM clause. In this example, MYDB is the connection alias.

```
proc sql;
  connect to SQLSVR as mydb
    (datasrc="SQL Server" user=myusr1 password=mypwd1);
  select * from connection to mydb
    (select CUSTOMER, NAME, COUNTRY
     from CUSTOMERS
     where COUNTRY <> 'USA');
quit;
```

This next example returns a list of the columns in the CUSTOMERS table.

```
proc sql;
  connect to SQLSVR as mydb
    (datasrc = "SQL Server" user=myusr1 password=mypwd1);
  select * from connection to mydb
    (ODBC::SQLColumns (, , "CUSTOMERS"));
quit;
```

## Passing Joins to Microsoft SQL Server

Microsoft SQL Server supports outer joins between two or more tables. Outer joins can be passed down to be processed in Microsoft SQL Server. However, the outer joins must not be mixed with inner joins in a query.

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

# Passing SAS Functions to Microsoft SQL Server

SAS/ACCESS Interface to Microsoft SQL Server passes the following SAS functions to the data source for processing if the DBMS server supports this function. Where the Microsoft SQL Server function name differs from the SAS function name, the Microsoft SQL Server name appears in parentheses. For details, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                          |                          |
|--------------------------|--------------------------|
| ABS                      | LOWCASE                  |
| ARCOS                    | MAX                      |
| ARSIN                    | MIN                      |
| ATAN                     | MINUTE                   |
| ATAN2                    | MOD (see note)           |
| AVG                      | MONTH                    |
| BYTE (CHAR)              | QTR (QUARTER)            |
| CEIL                     | REPEAT                   |
| COALESCE                 | SECOND                   |
| COS                      | SIGN                     |
| COSH                     | SIN                      |
| COT                      | SINH                     |
| COUNT                    | SOUNDEX                  |
| DAY (DAYOFMONTH)         | SQRT                     |
| DTEXTDAY (DAYOFMONTH)    | STD (STDEV)              |
| DTEXTMONTH (MONTH)       | STRIP (RTRIM with LTRIM) |
| DTEXTWEEKDAY (DAYOFWEEK) | SUBSTR (SUBSTRING)       |
| DTEXTYEAR (YEAR)         | SUM                      |
| EXP                      | TAN                      |
| FLOOR                    | TANH                     |
| HOUR                     | TRANWRD (REPLACE)        |
| INDEX (LOCATE)           | TRIMN (RTRIM)            |
| LEFT (LTRIM)             | UPCASE                   |
| LENGTH                   | VAR                      |
| LOG                      | WEEKDAY (DAYOFWEEK)      |
| LOG10                    | YEAR                     |

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

SQL\_FUNCTIONS=ALL can process SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. However, only when SQL\_FUNCTIONS=ALL can the SAS/ACCESS engine also pass these SAS SQL functions to Microsoft SQL Server. Because of incompatibility in date and time functions between Microsoft SQL Server and SAS, Microsoft SQL Server might not process them correctly. Check your results to determine whether these functions are working as expected.

|                    |                 |
|--------------------|-----------------|
| COMPRESS (REPLACE) | TIME (CURTIME)  |
| DATE (CURDATE)     | TODAY (CURDATE) |
| DATETIME (NOW)     |                 |

---

# DBLOAD Procedure Specifics for Microsoft SQL Server

---

## Overview

For general information about this feature, see the [DBLOAD Procedure on page 1455](#).

The Microsoft SQL Server under UNIX hosts interface supports all [DBLOAD procedure statements](#) (except ACCDESC=) in batch mode.

---

**Note:** SAS still supports this legacy procedure. However, to access your DBMS data more directly the best practice is to use the LIBNAME statement for Microsoft SQL Server or the SQL pass-through facility. For more information, see [“LIBNAME Statement for the Microsoft SQL Server Engine” on page 1009](#) and [“SQL Pass-Through Facility Specifics for Microsoft SQL Server” on page 1018](#).

---

Here are SAS/ACCESS Interface to Microsoft SQL Server specifics for the DBLOAD procedure.

- The DBLOAD step DBMS= value is `sqlsvr`.
- Here are the database description statements that PROC DBLOAD uses:

`DSN= <'>database-name<'>;`  
specifies the name of the database in which you want to store the new Microsoft SQL Server table. The *database-name* is limited to eight characters.

The database that you specify must already exist. If the database name contains the \_, \$, @, or # special character, you must enclose it in quotation marks. The Microsoft SQL Server standard recommends against using special characters in database names, however

`USER= <'>user name<'>;`  
enables you to connect to a Microsoft SQL Server database with a user ID that is different from the default ID.

USER= is optional in the Microsoft SQL Server interface. If you specify USER=, you must also specify PASSWORD=. If USER= is omitted, your default user ID is used.

PASSWORD=<'>*password*<'>;  
specifies the Microsoft SQL Server password that is associated with your user ID.

PASSWORD= is optional in the Microsoft SQL Server interface because users have default user IDs. If you specify USER=, you must specify PASSWORD=. If you do not wish to enter your SQL Server password in clear text on this statement, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

## Examples

This example creates a new Microsoft SQL Server table, MYUSR1.EXCHANGE, from the DLIB.RATEOFEX data file. You must be granted the appropriate privileges in order to create new Microsoft SQL Server tables or views.

```
proc dbload dbms=SQLSVR data=dlib.rateofex;
  dsn=sample; user='myusr1'; password='mypwd1'; table=exchange;
  rename fgnindol=fgnindollars
    4=dollarsinfgn;
  nulls updated=n fgnindollars=n
    dollarsinfgn=n country=n;
  load;
run;
```

This example only sends a Microsoft SQL Server SQL GRANT statement to the SAMPLE database and does not create a new table. Therefore, the TABLE= and LOAD statements are omitted.

```
proc dbload dbms=SQLSVR;
  user='myusr1';
  password='mypwd1';
  dsn=sample;
  sql grant select on myusr1.exchange
    to testcase;
run;
```

## Temporary Table Support for Microsoft SQL Server

SAS/ACCESS Interface to Microsoft SQL Server supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

# Locking in the Microsoft SQL Server Interface

These LIBNAME and data set options let you control how the Microsoft SQL Server interface handles locking. For details about options, see “[LIBNAME Options for Relational Databases](#)” on page 134.

`READ_LOCK_TYPE= ROW | TABLE | NOLOCK`

`UPDATE_LOCK_TYPE= ROW | TABLE | NOLOCK`

`READ_ISOLATION_LEVEL= S | RR | RC | RU | V`

The Microsoft SQL Server ODBC driver manager supports the S, RR, RC, RU, and V isolation levels, as defined in this table.

*Table 27.3 Isolation Levels for Microsoft SQL Server*

| Isolation Level       | Definition                                                                                                                                                                                                                      |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| S (serializable)      | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.                                                                                                                                                              |
| RR (repeatable read)  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.                                                                                                                                                    |
| RC (read committed)   | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.                                                                                                                                                    |
| RU (read uncommitted) | Allows dirty Reads, nonrepeatable Reads, and phantom Reads.                                                                                                                                                                     |
| V (versioning)        | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads. These transactions are serializable but higher concurrency is possible than with the serializable isolation level. Typically, a nonlocking protocol is used. |

Here is how the terms in the table are defined.

#### *Dirty read*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, the transaction can see changes that are made by those concurrent transactions even before they commit.

For example, if transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

*Nonrepeatable read*

If a transaction exhibits this phenomenon, it might read a row once and later fail when it attempts to read that row again in the same transaction. The row might have been changed or even deleted by a concurrent transaction. Therefore, the read is not necessarily repeatable.

For example, if transaction T1 retrieves a row, transaction T2 updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

*Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, transaction T1 retrieves the set of all rows that satisfy some condition. If transaction T2 inserts a new row that satisfies that same condition and transaction T1 repeats its retrieval request, it sees a row that did not previously exist, a *phantom*.

`UPDATE_ISOLATION_LEVEL= S | RR | RC | V`

The Microsoft SQL Server ODBC driver manager supports the S, RR, RC, and V isolation levels that are defined in the preceding table.

## Bulk Loading with Microsoft SQL Server

### Overview of Bulk Loading Setup

Bulk loading is supported for SAS/ACCESS Interface to Microsoft SQL Server. Bulk loading enables you to insert a large number of rows into a Microsoft SQL Server table. The setup steps and options that you use depend on your data source.

### Bulk Loading to Most Microsoft SQL Server Databases

You can enable bulk loading to most Microsoft SQL Server databases by specifying options in the odbc.ini file (for UNIX platforms) or by using the ODBC Administrator (for Windows platforms). For more information, see the ODBC driver information about bulk loading available at [Progress DataDirect](#).

To bulk load data to most Microsoft SQL Server databases, use the `INSERTBUFF=LIBNAME` option or `INSERTBUFF=` data set option to configure bulk loading with Microsoft SQL Server.

## Bulk Loading to Azure Synapse Analytics

When you bulk load data into Azure Synapse Analytics (SQL DW), SAS/ACCESS first loads the data files into an Azure Data Lake Storage (ADLS) Gen2 location. SAS/ACCESS then moves the files from the external storage account into Azure Synapse Analytics using the COPY INTO command.

**Note:** Bulk loading to Azure Synapse Analytics is supported only for SAS 9.4M7 and later. It is not supported in SAS Viya 3.5.

To bulk load data into Azure Synapse Analytics, use the AZUREAUTHCACHELOC= and AZURETENANTID= system options to control access to Azure. For details, see “[Authentication for Bulk Loading to Azure](#)”.

When you are bulk loading data into Azure Synapse Analytics, these options are available:

- [BL\\_ACCOUNTNAME= LIBNAME option and data set option](#)
- [BL\\_APPLICATIONID= LIBNAME option and data set option](#)
- [BL\\_COMPRESS= LIBNAME option and data set option](#)
- [BL\\_DEFAULT\\_DIR= LIBNAME option and data set option](#)
- [BL\\_DELETE\\_DATAFILE= LIBNAME option and data set option](#)
- [BL\\_DELIMITER= LIBNAME option and data set option](#)
- [BL\\_DNSSUFFIX= LIBNAME option and data set option](#)
- [BL\\_FILESYSTEM= LIBNAME option and data set option](#)
- [BL\\_FOLDER= LIBNAME option and data set option](#)
- [BL\\_IDENTITY= LIBNAME option and data set option](#)
- [BL\\_LOG= LIBNAME option and data set option](#)
- [BL\\_MAXERRORS= LIBNAME option and data set option](#)
- [BL\\_NUM\\_DATAFILES= data set option](#)
- [BL\\_OPTIONS= LIBNAME option and data set option](#)
- [BL\\_SECRET= LIBNAME option and data set option](#)
- [BL\\_TIMEOUT= LIBNAME option and data set option](#)
- [BL\\_USE\\_ESCAPE= LIBNAME option and data set option](#)
- [BL\\_USE\\_LOG= LIBNAME option and data set option](#)
- [BULKLOAD= LIBNAME option and data set option](#)

---

## Authentication for Bulk Loading to Azure

For bulk loading, additional authentication is required due to the process that is described in “[Bulk Loading to Azure Synapse Analytics](#)” on page 1025.

Use device-code authentication to access Azure. The user ID that is used to authenticate must be assigned Contributor and Storage Blob Data Contributor roles in the ADLS account configuration.

- Use the following system options:
  - [AZURETENANTID=](#) is required.
  - [AZUREAUTHCACHELOC=](#) is optional.
- The following options are all required, either as LIBNAME or data set options:
  - BL\_ACCOUNTNAME=
  - BL\_APPLICATIONID=
  - BL\_FILESYSTEM=
  - BL\_IDENTITY=
  - BL\_SECRET=
- On your first attempt to access data by using device-code authentication, an error message is printed in the SAS log. You must authenticate with Microsoft. That process is explained in the documentation for the required [AZURETENANTID=](#) system option.

---

## Naming Conventions for Microsoft SQL Server

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Microsoft SQL Server is not case sensitive, so all names default to mixed case.

Microsoft SQL Server supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, SAS truncates them to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

# Data Types for Microsoft SQL Server

## Overview

Every column in a table has a name and a data type. The data type tells the Microsoft SQL Server how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Microsoft SQL Server [null and default values](#) and [data conversions](#).

## Microsoft SQL Server Null Values

Microsoft SQL Server has a special value called NULL. A Microsoft SQL Server NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Microsoft SQL Server NULL value, it interprets it as a SAS missing value.

Microsoft SQL Server columns can be specified as NOT NULL so that they require data—they cannot contain NULL values. When a column is specified as NOT NULL, the DBMS does not add a row to the table unless the row has a value for that column. When creating a DBMS table with SAS/ACCESS, you can use the [DBNULL=](#) data set option to indicate whether NULL is a valid value for specified columns.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how SAS missing character values are handled, use the [NULLCHAR=](#) and [NULLCHARVAL=](#) data set options.

## LIBNAME Statement Data Conversions

This table shows all data types that SAS/ACCESS Interface to Microsoft SQL Server supports. This table also shows the default SAS format that corresponds to each data type.

*Table 27.4 Microsoft SQL Server Data Types and Default SAS Formats*

| Microsoft SQL Server Data Type | Default SAS Format |
|--------------------------------|--------------------|
| CHAR( <i>n</i> )               | \$ <i>w.</i>       |
| NCHAR( <i>n</i> )              | \$ <i>w.</i>       |

| Microsoft SQL Server Data Type | Default SAS Format                                                                                          |
|--------------------------------|-------------------------------------------------------------------------------------------------------------|
| VARCHAR( <i>n</i> )            | \$ <i>w</i> .                                                                                               |
| NVARCHAR( <i>n</i> )           | \$ <i>w</i> .                                                                                               |
| TEXT                           | \$1024.                                                                                                     |
| NTEXT                          | \$1024.                                                                                                     |
| UNIQUEIDENTIFIER               | \$36.                                                                                                       |
| BINARY( <i>n</i> )             | \$ <i>w</i> . <sup>1</sup>                                                                                  |
| VARBINARY( <i>n</i> )          | \$ <i>w</i> . <sup>1</sup>                                                                                  |
| VARBINARY( <i>max</i> )        | \$ <i>w</i> . <sup>1</sup>                                                                                  |
| DECIMAL( <i>p</i> , <i>s</i> ) | <i>w</i> . or <i>w.d</i> or none if <i>w</i> and <i>d</i> are not specified                                 |
| NUMERIC( <i>p</i> , <i>s</i> ) | <i>w</i> . or <i>w.d</i> or none if <i>w</i> and <i>d</i> are not specified                                 |
| BIGINT                         | 20.                                                                                                         |
| INT                            | 11.                                                                                                         |
| SMALLINT                       | 6.                                                                                                          |
| TINYINT                        | 4.                                                                                                          |
| BIT                            | 1.                                                                                                          |
| FLOAT(24)                      | none                                                                                                        |
| FLOAT( <i>n</i> )              | none                                                                                                        |
| FLOAT(53)                      | none                                                                                                        |
| DATE                           | DATE9.                                                                                                      |
| TIME( <i>n</i> )               | TIME8.<br>SAS/ACCESS Interface to Microsoft SQL Server cannot support fractions of seconds for time values. |
| DATETIME                       | DATETIME <i>w.d</i> where <i>w</i> and <i>d</i> depend on precision                                         |

| Microsoft SQL Server Data Type | Default SAS Format                            |
|--------------------------------|-----------------------------------------------|
| DATETIME2                      | DATETIMEw.d where w and d depend on precision |
| SMALLDATETIME                  | DATETIMEw.d where w and d depend on precision |
| DATETIMEOFFSET                 | DATETIMEw.d where w and d depend on precision |
| MONEY                          | 21.4                                          |
| SMALLMONEY                     | 12.4                                          |

**1** Because the Microsoft SQL Server driver does the conversion, this field is displayed as if the \$HEXw. format were applied.

This table shows the default data types that the Microsoft SQL Server interface uses when creating tables.

**Table 27.5 Default Microsoft SQL Server Output Data Types**

| SAS Variable Format | Default Microsoft SQL Server Data Type                                        |
|---------------------|-------------------------------------------------------------------------------|
| w.d                 | DOUBLE(m.n) or NUMERIC(m.n) <sup>2</sup>                                      |
| \$w.                | VARCHAR(n) <sup>1</sup>                                                       |
| Datetime formats    | DATETIME                                                                      |
| Date formats        | DATE                                                                          |
| Time formats        | TIME(n)                                                                       |
| DOLLARw.d           | MONEYor SMALLMONEY<br>See “ <a href="#">Conversion to Money Data Types</a> ”. |
| \$HEXw.             | VARBINARY(n)                                                                  |

**1** n in Microsoft SQL Server character data types is equivalent to w in SAS formats.

**2** m and n in Microsoft SQL Server numeric data types are equivalent to w and d in SAS formats.

The Microsoft SQL Server interface allows non-default data types to be specified with the [DBTYPE=](#) on page 536 data set option.

---

## Conversion to Money Data Types

Whether SAS/ACCESS converts data in a DOLLARw.d format to MONEY or SMALLMONEY in Microsoft SQL Server depends on the length and precision that are used in the DOLLARw.d format. If the difference between the length and the precision is 6 or less, then the data values are converted to SMALLMONEY when the data is written to Microsoft SQL Server. If the difference is 7 or larger, then the data values are converted to the MONEY data type. For example, a column that is formatted as DOLLAR10.4 in SAS would be converted to SMALLMONEY in Microsoft SQL Server, because the difference between the length (10) and the precision (4) is 6.

*Table 27.6 Examples Conversions from SAS Formats to Money Data Types in Microsoft SQL Server*

| SAS Format | Data Type in Microsoft SQL Server |
|------------|-----------------------------------|
| DOLLAR6.2  | SMALLMONEY                        |
| DOLLAR6.1  | SMALLMONEY                        |
| DOLLAR6.   | SMALLMONEY                        |
| DOLLAR7.2  | SMALLMONEY                        |
| DOLLAR7.1  | SMALLMONEY                        |
| DOLLAR7.   | MONEY                             |
| DOLLAR9.3  | SMALLMONEY                        |
| DOLLAR9.2  | MONEY                             |
| DOLLAR9.1  | MONEY                             |
| DOLLAR9.   | MONEY                             |

---

## Sample Programs for Microsoft SQL Server

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to MySQL

|                                                                    |      |
|--------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to MySQL</i> ..... | 1032 |
| <i>Introduction to SAS/ACCESS Interface to MySQL</i> .....         | 1032 |
| <i>LIBNAME Statement for the MySQL Engine</i> .....                | 1033 |
| Overview .....                                                     | 1033 |
| Arguments .....                                                    | 1033 |
| MySQL LIBNAME Statement Example .....                              | 1036 |
| <i>Data Set Options for MySQL</i> .....                            | 1036 |
| <i>SQL Pass-Through Facility Specifics for MySQL</i> .....         | 1038 |
| Key Information .....                                              | 1038 |
| Examples .....                                                     | 1038 |
| <i>Autocommit and Table Types</i> .....                            | 1039 |
| <i>Understanding MySQL Update and Delete Rules</i> .....           | 1040 |
| <i>Temporary Table Support for MySQL</i> .....                     | 1040 |
| <i>Passing SAS Functions to MySQL</i> .....                        | 1041 |
| <i>Passing Joins to MySQL</i> .....                                | 1042 |
| <i>Bulk Loading for MySQL</i> .....                                | 1042 |
| Overview .....                                                     | 1042 |
| Data Set Options for Bulk Loading .....                            | 1042 |
| Examples .....                                                     | 1043 |
| <i>Naming Conventions for MySQL</i> .....                          | 1043 |
| <i>Case Sensitivity for MySQL</i> .....                            | 1044 |
| <i>Data Types for MySQL</i> .....                                  | 1044 |
| Overview .....                                                     | 1044 |
| Supported MySQL Data Types .....                                   | 1045 |
| LIBNAME Statement Data Conversions .....                           | 1045 |
| <i>Sample Programs for MySQL</i> .....                             | 1048 |

---

# System Requirements for SAS/ACCESS Interface to MySQL

You can find information about system requirements for SAS/ACCESS Interface to MySQL in the following locations:

- [System Requirements for SAS/ACCESS Interface to MySQL with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to MySQL

For available SAS/ACCESS features, see [MySQL supported features on page 108](#). For more information about MySQL, see your MySQL documentation.

---

**Note:** Beginning with SAS 9.4M8, SAS/ACCESS Interface to MySQL on AIX is no longer available. If you have an existing instance of SAS/ACCESS Interface to MySQL on AIX and plan to upgrade to SAS 9.4M8 or later, SAS recommends that you first unconfigure and uninstall it. For more information, see [Unconfiguring and Uninstalling Retired Products](#).

---

When you use SAS/ACCESS Interface to MySQL to access a SingleStore data source, ensure that your session encoding is set to UTF-8. For more information, see “[ENCODING System Option: UNIX, Windows, and z/OS](#)” in [SAS National Language Support \(NLS\): Reference Guide](#).

SAS/ACCESS Interface to MySQL includes SAS Data Connector to MySQL. The data connector enables you to transfer large amounts of data between your MySQL database and the CAS server for parallel processing. For more information see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“MySQL Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

---

# LIBNAME Statement for the MySQL Engine

---

## Overview

This section describes the LIBNAME statements that SAS/ACCESS Interface to MySQL supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing MySQL.

**LIBNAME** *libref mysql* <*connection-options*><*LIBNAME-options*>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables.

*mysql*

specifies the SAS/ACCESS engine name for MySQL interface.

*connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are defined.

---

**Note:** All of these connection options (preceding the list of MySQL LIBNAME statement options) are valid when used in the CONNECT statement with the SQL procedure.

---

**USER=<'>MySQL-user name<'>**

specifies the MySQL user logon ID. If this argument is not specified, the current user is assumed. If the user name contains spaces or nonalphanumeric characters, you must enclose the user name in quotation marks.

**PASSWORD=<'>MySQL-password<'>**

specifies the MySQL password that is associated with the MySQL logon ID. If the password contains spaces or nonalphanumeric characters, you must enclose the password in quotation marks.

**DATABASE=<'>MySQL-database<'>**

specifies the MySQL database to which you want to connect. If the database name contains spaces or nonalphanumeric characters, you must enclose the database name in quotation marks.

**SERVER=<'>MySQL-server<'>**

specifies the server name or IP address of the MySQL server. If the server name contains spaces or nonalphanumeric characters, you must enclose the server name in quotation marks.

**PORT=port**

specifies the port used to connect to the specified MySQL server.

Default: 3306

**LIBNAME-options**

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to MySQL, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For more information, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 28.1** SAS/ACCESS LIBNAME Options for MySQL

| Option             | Default Value                                                                                   | Valid in CONNECT |
|--------------------|-------------------------------------------------------------------------------------------------|------------------|
| ACCESS=            | none                                                                                            |                  |
| AUTHDOMAIN=        | none                                                                                            | •                |
| AUTOCOMMIT=        | YES                                                                                             | •                |
| BL_DEFAULT_DIR=    | temporary file directory that is specified by the UTILLOC= system option                        | •                |
| BULKLOAD           | NO                                                                                              | •                |
| CONNECTION=        | SHAREDREAD                                                                                      | •                |
| CONNECTION_GROUP=  | none                                                                                            | •                |
| DBCLIENT_MAX_BYTES | SAS session encoding                                                                            |                  |
| DBCOMMIT=          | 1000 (when inserting rows), 0 (when updating, deleting, or appending rows to an existing table) |                  |
| DBCONINIT=         | none                                                                                            | •                |

| Option               | Default Value         | Valid in CONNECT |
|----------------------|-----------------------|------------------|
| DBCONTERM=           | none                  | •                |
| DBCREATE_TABLE_OPTS= | none                  |                  |
| DBGEN_NAME=          | DBMS                  | •                |
| DBINDEX=             | NO                    |                  |
| DBLIBINIT=           | none                  |                  |
| DBLIBTERM=           | none                  |                  |
| DBMAX_TEXT=          | 1024                  | •                |
| DBMSTEMP=            | NO                    |                  |
| DBPROMPT=            | NO                    | •                |
| DBSASLABEL=          | COMPAT                |                  |
| DEFAULT_AUTH_PLUGIN= | caching_sha2_password |                  |
| DEFER=               | NO                    | •                |
| DIRECT_EXE=          | none                  |                  |
| DIRECT_SQL=          | YES                   |                  |
| ESCAPE_BACKSLASH=    | NO                    |                  |
| INSERTBUFF=          | 1                     |                  |
| MULTI_DATASRC_OPT=   | NONE                  |                  |
| PRESERVE_COL_NAMES=  | YES                   |                  |
| PRESERVE_TAB_NAMES=  | YES                   |                  |
| QUALIFIER=           | none                  |                  |
| REREAD_EXPOSURE=     | NO                    | •                |
| RESULTS=             | MEMORY                |                  |
| SCHEMA               | DBMS specific         |                  |

| Option              | Default Value | Valid in CONNECT |
|---------------------|---------------|------------------|
| SPOOL=              | YES           |                  |
| SQL_FUNCTIONS=      | none          |                  |
| SQL_FUNCTIONS_COPY= | none          |                  |
| SSL_CA=             | none          |                  |
| SSL_CERT=           | none          |                  |
| SSL_CIPHER=         | none          |                  |
| SSL_KEY=            | none          |                  |
| UTILCONN_TRANSIENT= | NO            |                  |

---

## MySQL LIBNAME Statement Example

In the following example, the libref MYSQLLIB uses SAS/ACCESS Interface to MySQL to connect to a MySQL database. The SAS/ACCESS connection options are USER=, PASSWORD=, DATABASE=, SERVER=, and PORT=.

```
libname mysqllib mysql user=myusr1 password=mypwd1 database=mysqlldb
server=mysrv1 port=9876;

proc print data=mysqllib.employees;
  where dept='CSR010';
run;
```

---

## Data Set Options for MySQL

All SAS/ACCESS data set options in this table are supported for MySQL. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 28.2** Data Set Options for MySQL

| Option               | Default Value                                                            |
|----------------------|--------------------------------------------------------------------------|
| BL_DEFAULT_DIR=      | temporary file directory that is specified by the UTILLOC= system option |
| BULKLOAD=            | NO                                                                       |
| DBCOMMIT=            | LIBNAME option value                                                     |
| DBCONDITION=         | none                                                                     |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                                                     |
| DBGEN_NAME=          | DBMS                                                                     |
| DBINDEX=             | LIBNAME option value                                                     |
| DBKEY=               | none                                                                     |
| DBLABEL=             | NO                                                                       |
| DBLARGETABLE=        | none                                                                     |
| DBMAX_TEXT=          | 1024                                                                     |
| DBNULL=              | YES                                                                      |
| DBPROMPT=            | LIBNAME option value                                                     |
| DBSASLABEL=          | COMPAT                                                                   |
| DBSASTYPE=           | see “Data Types for MySQL”                                               |
| DBTYPE=              | see “LIBNAME Statement Data Conversions”                                 |
| ESCAPE_BACKSLASH=    | NO                                                                       |
| INSERTBUFF=          | 1                                                                        |
| NULCHAR=             | SAS                                                                      |
| NULCHARVAL=          | a blank character                                                        |
| PRESERVE_COL_NAMES=  | LIBNAME option value                                                     |
| QUALIFIER=           | LIBNAME option value                                                     |
| SASDATEFMT=          | DATETIME20.0                                                             |

| Option                  | Default Value        |
|-------------------------|----------------------|
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |

---

## SQL Pass-Through Facility Specifics for MySQL

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for MySQL.

- The *dbms-name* is `mysql`.
- If you call MySQL stored procedures that return multiple result sets, SAS returns only the last result set.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).

**Note:** Due to a current limitation in the MySQL client library, you cannot run MySQL stored procedures when SAS is running on AIX.

---

### Examples

This example uses the alias DBCON for the DBMS connection (the connection alias is optional):

```
proc sql;
  connect to mysql as dbcon
    (user=myusr1 password=mypwd1 server=mysrv1
     database=mysqldb port=9876);
quit;
```

This example connects to MySQL and sends it two EXECUTE statements to process:

```
proc sql;
  connect to mysql (user=myusr1 password=mypwd1 server=mysrv1
    database=mysqldb port=9876);
  execute (create table whotookorders as
```

```

select ordernum, takenby,
       firstname, lastname, phone
  from orders, employees
 where orders.takenby=employees.empid)
by mysql;
execute (grant select on whotookorders
           to myusr1) by mysql;
disconnect from mysql;
quit;

```

This example performs a query, shown in highlighted text, on the MySQL table CUSTOMERS:

```

proc sql;
connect to mysql (user=myusr1 password=mypwd1 server=mysrv1
                  database=mysql2 port=9876);
select *
  from connection to mysql
 (select * from customers
   where customer like '1%');
disconnect from mysql;
quit;

```

## Autocommit and Table Types

MySQL supports several table types, two of which are InnoDB (the default) and MyISAM. A single database can contain tables of different types. The behavior of a table is determined by its table type. For example, by definition, a table created of MyISAM type does not support transactions. Consequently, all DML statements (updates, deletes, inserts) are automatically committed. If you need transactional support, specify a table type of InnoDB in the **DBCREATE\_TABLE\_OPTS=LIBNAME** option. This table type allows for updates, deletes, and inserts to be rolled back if an error occurs; or updates, deletes, and inserts to be committed if the SAS DATA step or procedure completes successfully.

By default, the MYSQL LIBNAME engine sets **AUTOCOMMIT=YES** regardless of the table type. If you are using tables of the type InnoDB, set **AUTOCOMMIT=NO** to improve performance. To control how often COMMITS are executed, specify the **DBCOMMIT=** option.

---

**Note:** The DBCOMMIT option can affect SAS/ACCESS performance. Experiment with a value that best fits your table size and performance needs before using it for production jobs. Transactional tables require significantly more memory and disk space requirements.

---

---

# Understanding MySQL Update and Delete Rules

To avoid data integrity problems when updating or deleting data, you need to specify a primary key on your table. See MySQL documentation for more information about table types and transactions.

This example uses AUTOCOMMIT=NO and DBTYPE= to create the primary key and also DBCREATE\_TABLE\_OPTS= to determine the MySQL table type.

```
libname invty mysql user=myusr1 server=mysrv1 database=test autocommit=no
reread_exposure=no;

proc sql;
drop table invty.STOCK23;
quit;

/* Create DBMS table with primary key and of type INNODB */
data invty.STOCK23(drop=PARTNO DBTYPE=(RECDATE="date not null,
primary key(RECDATE)") DBCREATE_TABLE_OPTS="engine = innodb");
input PARTNO $ DESCX $ INSTOCK @20
      RECDATE date7. @29 PRICE;
format RECDATE date7.;
datalines;
K89R seal     34 27jul95 245.00
M447 sander   98 20jun95 45.88
LK43 filter   121 19may96 10.99
MN21 brace    43 10aug96 27.87
BC85 clamp    80 16aug96  9.55
KJ66 cutter    6 20mar96 24.50
UYN7 rod      211 18jun96 19.77
JD03 switch   383 09jan97 13.99
BV1I timer    26 03jan97 34.50
;
```

This next example shows how you can update the table now that STOCK23 has a primary key.

```
proc sql;
update invty.STOCK23 set price=price*1.1 where INSTOCK > 50;
quit;
```

---

# Temporary Table Support for MySQL

SAS/ACCESS Interface to MySQL supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

# Passing SAS Functions to MySQL

`SQL_FUNCTIONS=ALL` allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when `SQL_FUNCTIONS=ALL` can the SAS/ACCESS engine also pass these SAS SQL functions to MySQL. Due to incompatibility in date and time functions between MySQL and SAS, MySQL might not process them correctly. Check your results to determine whether these functions are working as expected.

Where the MySQL function name differs from the SAS function name, the MySQL name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

By default, the following SAS functions are passed down to the database:

|                  |                     |
|------------------|---------------------|
| ABS              | LOWCASE (LCASE)     |
| ARCOS (ACOS)     | MAX                 |
| ARSIN (ASIN)     | MIN                 |
| ATAN             | MINUTE              |
| ATAN2            | MOD (see note)      |
| AVG              | MONTH               |
| CEIL (CEILING)   | QTR (QUARTER)       |
| COALESCE         | SECOND              |
| COS              | SIGN                |
| COT              | SIN                 |
| COUNT            | SQRT                |
| DAY (DAYOFMONTH) | STRIP (TRIM)        |
| EXP              | TAN                 |
| FLOOR            | TRANWRD (REPLACE)   |
| HOUR             | TRIMN (RTRIM)       |
| INDEX (LOCATE)   | UPCASE (UCASE)      |
| LOG              | WEEKDAY (DAYOFWEEK) |
| LOG2             | YEAR                |
| LOG10            |                     |

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

The following SAS functions are passed down to the database only if `SQL_FUNCTIONS=ALL`:

|                    |                    |
|--------------------|--------------------|
| BYTE (CHAR)        | ROUND              |
| COMPRESS (REPLACE) | SOUNDEX            |
| DATE (CURDATE)     | SUBSTR (SUBSTRING) |

|                |                 |
|----------------|-----------------|
| DATEPART       | TIME (CURTIME)  |
| DATETIME (NOW) | TIMEPART        |
| LENGTH         | TODAY (CURDATE) |
| REPEAT         |                 |

## Passing Joins to MySQL

For a multiple libref join to pass to MySQL, all of these components of the LIBNAME statements must match exactly:

- user (USER=)
- password (PASSWORD=)
- database (DATABASE=)
- server (SERVER=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

## Bulk Loading for MySQL

### Overview

Bulk loading is the fastest way to insert large numbers of rows into a MySQL table. Using this facility instead of regular SQL insert statements, you can insert rows more rapidly.

The MySQL LIBNAME engine calls the MySQL bulk-load facility when you specify BULKLOAD=YES. You must specify **BULKLOAD=YES** to use this facility.

This facility uses the LOAD DATA command to create a text file on the client, which lets MySQL bulk load it into a table on the server.

## Data Set Options for Bulk Loading

Here are the MySQL bulk-load data set options. For detailed information about these options, see [Data Set Options for Relational Databases](#) on page 370.

- [BL\\_DEFAULT\\_DIR=](#)
- [BULKLOAD=](#)

---

## Examples

This example uses the DATA step.

```
libname x mysql user=myusr1 server=mysrv1 database=mydb1 password=mypwd1;

data x.mydata_bulk21(BL_DEFAULT_DIR="c:/temp/mysql" bulkload=yes);
  col1=1;col2='Test';
run;
```

This next example uses PROC APPEND.

```
libname x mysql user=myusr1 server=mysrv1 database=mydb1 password=mypwd1;

data x;col1=1;col2='TEST';
output;
run;

proc append data=x base=x.mydata_bulk21(bulkload=yes);
run;
```

---

## Naming Conventions for MySQL

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

MySQL database identifiers that you can name include databases, tables, and columns. They follow these naming conventions.

- Aliases must be from 1 to 255 characters long. All other identifier names must be from 1 to 64 characters long.
- Database names can use any character that is allowed in a directory name except for a period, a backward slash (\), or a forward slash (/).
- By default, MySQL encloses column names and table names in quotation marks.
- Table names can use any character that is allowed in a file name except for a period or a forward slash.
- Table names must be 32 characters or less because SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.
- Column names and alias names allow all characters.
- Embedded spaces and other special characters are not permitted unless you enclose the name in quotation marks.
- Embedded quotation marks are not permitted.
- Case sensitivity is specified when a server is installed. By default, the names of database objects are case sensitive on UNIX and not case sensitive on

Windows. For example, the names CUSTOMER and Customer are different on a case-sensitive server.

- A name cannot be a reserved word in MySQL unless you enclose the name in quotation marks. See the MySQL documentation for more information about reserved words.
- Database names must be unique. For each user within a database, names of database objects must be unique across all users. For example, if a database contains a department table that User A created, no other user can create a department table in the same database.

MySQL does not recognize the notion of schema, so tables are automatically visible to all users with the appropriate privileges. Column names and index names must be unique within a table.

---

## Case Sensitivity for MySQL

In MySQL, databases and tables correspond to directories and files within those directories. Consequently, the case sensitivity of the underlying operating system determines the case sensitivity of database and table names. This means database and table names are not case sensitive in Windows, and case sensitive in most varieties of UNIX.

In SAS, names can be entered in either uppercase or lowercase. MySQL recommends that you adopt a consistent convention of either all uppercase or all lowercase table names, especially on UNIX hosts. This can be easily implemented by starting your server with `-O lower_case_table_names=1`. Please see the MySQL documentation for more details.

If your server is on a case-sensitive platform and you choose to allow case sensitivity, be aware that when you reference MySQL objects through the SAS/ACCESS interface, objects are case sensitive and require no quotation marks. Also, in the SQL pass-through facility, all MySQL object names are case sensitive. Names are passed to MySQL exactly as they are entered.

For more information about case sensitivity and MySQL names, see “[Naming Conventions for MySQL](#)” on page 1043.

---

## Data Types for MySQL

---

### Overview

Every column in a table has a name and a data type. The data type tells MySQL how much physical storage to set aside for the column and the form in which the

data is stored. This section includes information about MySQL data types and data conversions.

## Supported MySQL Data Types

Here are the data types that the MySQL engine supports.

- Character data:

|                                         |                                         |
|-----------------------------------------|-----------------------------------------|
| BLOB (binary large object)              | MEDIUMTEXT                              |
| CHAR ( <i>n</i> )                       | SET ("value1", "value2", "value3", ...) |
| ENUM ("value1", "value2", "value3",...) | TEXT                                    |
| JSON                                    | TINYBLOB                                |
| LONGBLOB                                | TINYTEXT                                |
| LONGTEXT                                | VARCHAR ( <i>n</i> )                    |
| MEDIUMBLOB                              |                                         |

- Numeric data:

|                                     |                        |
|-------------------------------------|------------------------|
| BIGINT ( <i>n</i> )                 | INT ( <i>n</i> )       |
| DECIMAL ( <i>length, decimals</i> ) | MEDIUMINT ( <i>n</i> ) |
| DOUBLE ( <i>length, decimals</i> )  | SMALLINT ( <i>n</i> )  |
| FLOAT ( <i>length, decimals</i> )   | TINYINT ( <i>n</i> )   |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Date, time, and timestamp data:

|          |           |
|----------|-----------|
| DATE     | TIME      |
| DATETIME | TIMESTAMP |

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to MySQL assigns to SAS variables when using the [LIBNAME statement](#) to read from a MySQL table. These default formats are based on MySQL column attributes.

**Table 28.3** LIBNAME Statement: Default SAS Formats for MySQL Data Types

| MySQL Column Type                | SAS Data Type | Default SAS Format         |
|----------------------------------|---------------|----------------------------|
| CHAR( <i>n</i> ) <sup>1</sup>    | character     | \$ <i>w</i> .              |
| VARCHAR( <i>n</i> ) <sup>1</sup> | character     | \$ <i>w</i> .              |
| TINYTEXT                         | character     | \$ <i>w</i> .              |
| TEXT                             | character     | \$ <i>w</i> . <sup>2</sup> |
| MEDIUMTEXT                       | character     | \$ <i>w</i> . <sup>2</sup> |
| LONGTEXT                         | character     | \$ <i>w</i> . <sup>2</sup> |
| JSON                             | character     | \$ <i>w</i> . <sup>2</sup> |
| TINYBLOB                         | character     | \$ <i>w</i> . <sup>2</sup> |
| BLOB                             | character     | \$ <i>w</i> . <sup>2</sup> |
| MEDIUMBLOB                       | character     | \$ <i>w</i> . <sup>2</sup> |
| LONGBLOB                         | character     | \$ <i>w</i> . <sup>2</sup> |
| ENUM                             | character     | \$ <i>w</i> .              |
| SET                              | character     | \$ <i>w</i> .              |
| TINYINT                          | numeric       | 4.0                        |
| SMALLINT                         | numeric       | 6.0                        |
| MEDIUMINT                        | numeric       | 8.0                        |
| INT                              | numeric       | 11.0                       |
| BIGINT                           | numeric       | 20.                        |
| DECIMAL                          | numeric       | <i>w.d</i>                 |
| FLOAT                            | numeric       |                            |
| DOUBLE                           | numeric       |                            |
| DATE                             | numeric       | DATE                       |
| TIME                             | numeric       | TIME                       |

| MySQL Column Type | SAS Data Type | Default SAS Format |
|-------------------|---------------|--------------------|
| DATETIME          | numeric       | DATETIME           |
| TIMESTAMP         | numeric       | DATETIME           |

**1** *n* in MySQL character data types is equivalent to *w* in SAS formats.

**2** In this case, *w* is the value of the DBMAX\_TEXT= option.

This table shows the default MySQL data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

*Table 28.4 LIBNAME Statement: Default MySQL Data Types for SAS Variable Formats*

| SAS Variable Format                    | MySQL Data Type                                    |
|----------------------------------------|----------------------------------------------------|
| <i>w.d</i> <sup>1</sup>                | DECIMAL ([ <i>m</i> -1], <i>n</i> ) <sup>2,3</sup> |
| <i>w.</i> (where <i>w</i> <= 2)        | TINYINT                                            |
| <i>w.</i> (where <i>w</i> <= 4)        | SMALLINT                                           |
| <i>w.</i> (where <i>w</i> <= 6)        | MEDIUMINT                                          |
| <i>w</i> (where <i>w</i> <= 17)        | BIGINT                                             |
| other numerics                         | DOUBLE                                             |
| \$ <i>w.</i> (where <i>w</i> <= 65535) | VARCHAR( <i>n</i> ) <sup>1</sup>                   |
| \$ <i>w.</i> (where <i>w</i> > 65535)  | TEXT                                               |
| datetime formats                       | TIMESTAMP                                          |
| date formats                           | DATE                                               |
| time formats                           | TIME                                               |

**1** *n* in MySQL character data types is equivalent to *w* in SAS formats.

**2** *m* and *n* in MySQL numeric data types are equivalent to *w* and *d* in SAS formats.

**3** DECIMAL types are created as (*m*-1, *n*). SAS includes space to write the value, the decimal point, and a minus sign (if necessary) in its calculation for precision. These must be removed when converting to MySQL.

---

## Sample Programs for MySQL

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to Netezza

|                                                                      |      |
|----------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to Netezza</i> ..... | 1050 |
| <i>Introduction to SAS/ACCESS Interface to Netezza</i> .....         | 1050 |
| <b>LIBNAME Statement for the Netezza Engine</b> .....                | 1050 |
| Overview .....                                                       | 1050 |
| Arguments .....                                                      | 1051 |
| Netezza LIBNAME Statement Examples .....                             | 1054 |
| <b>Data Set Options for Netezza</b> .....                            | 1055 |
| <b>SQL Pass-Through Facility Specifics for Netezza</b> .....         | 1057 |
| Key Information .....                                                | 1057 |
| CONNECT Statement Example .....                                      | 1057 |
| Special Catalog Queries .....                                        | 1058 |
| <b>Temporary Table Support for Netezza</b> .....                     | 1059 |
| <b>Passing SAS Functions to Netezza</b> .....                        | 1059 |
| <b>Passing Joins to Netezza</b> .....                                | 1060 |
| <b>Bulk Loading and Unloading for Netezza</b> .....                  | 1061 |
| Loading .....                                                        | 1061 |
| Unloading .....                                                      | 1062 |
| <b>Naming Conventions for Netezza</b> .....                          | 1063 |
| <b>Data Types for Netezza</b> .....                                  | 1064 |
| Overview .....                                                       | 1064 |
| Supported Netezza Data Types .....                                   | 1064 |
| Loading Large Numeric Values .....                                   | 1065 |
| Default SAS Formats for Netezza NUMERIC Data Types .....             | 1065 |
| Netezza Null Values .....                                            | 1066 |
| LIBNAME Statement Data Conversions .....                             | 1066 |
| <b>Sample Programs for Netezza</b> .....                             | 1068 |

---

# System Requirements for SAS/ACCESS Interface to Netezza

You can find information about system requirements for SAS/ACCESS Interface to Netezza in the following locations:

- [System Requirements for SAS/ACCESS Interface to Netezza with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to Netezza

For available SAS/ACCESS features, see [Netezza supported features on page 109](#).  
For more information about Netezza, see your Netezza documentation.

---

# LIBNAME Statement for the Netezza Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Netezza supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Netezza.

**LIBNAME** *libref* **netezza** <connection-options> <LIBNAME-options>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *netezza*

specifies the SAS/ACCESS engine name for the Netezza interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the Netezza Performance Server in one of two ways. Specify only one of these methods for each connection because they are mutually exclusive.

- SERVER=, DATABASE=, PORT=, USER=, PASSWORD=, READ\_ONLY=
- DATASRC=, USER=, PASSWORD=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

#### **SERVER=<'>server-name<'>**

specifies the server name or IP address of the Netezza Performance Server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters or if it is an IP address, you must enclose it in quotation marks.

#### **DATABASE=<'>database-name<'>**

specifies the name of the database on the Netezza Performance Server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

#### **PORT=port**

specifies the port number that is used to connect to the specified Netezza Performance Server. If you do not specify a port, the default is 5480.

#### **USER=<'>Netezza-user-name<'>**

specifies the Netezza user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

#### **PASSWORD=<'>Netezza-password<'>**

specifies the password that is associated with your Netezza user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: PASS=, PW=, PWD=

**READ\_ONLY=YES | NO**

specifies whether to connect to the Netezza database in Read-Only mode (YES) or read-write (NO) mode. If you do not specify anything for **READ\_ONLY=**, the default of NO is used.

Alias: **READONLY=**

**DATASRC=<'>Netezza-data-source<'>**

specifies the configured Netezza ODBC data source to which you want to connect. Use this option if you have existing Netezza ODBC data sources that are configured on your client. This method requires additional setup—either through the ODBC Administrator control panel on Windows platforms or through the odbc.ini file or a similarly named configuration file on UNIX platforms. It is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

Aliases: **DSN=**, **DS=**

#### *LIBNAME-options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Netezza, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For more information, see [LIBNAME Options for Relational Databases on page 134](#) or “SQL Pass-Through Facility” on page 689.

**Table 29.1** SAS/ACCESS LIBNAME Options for Netezza

| Option                   | Default Value                                  | Valid in CONNECT |
|--------------------------|------------------------------------------------|------------------|
| <b>ACCESS=</b>           | none                                           |                  |
| <b>AUTHDOMAIN=</b>       | none                                           |                  |
| <b>AUTOCOMMIT=</b>       | operation-specific                             | •                |
| <b>BULKUNLOAD=</b>       | NO                                             | •                |
| <b>CHAR_AS_NCHAR=</b>    | NO                                             |                  |
| <b>CONNECTION=</b>       | UNIQUE                                         | •                |
| <b>CONNECTION_GROUP=</b> | none                                           | •                |
| <b>CONOPTS=</b>          | none                                           | •                |
| <b>DBCOMMIT=</b>         | 1000 when inserting rows, 0 when updating rows |                  |
| <b>DBCONINIT=</b>        | none                                           | •                |
| <b>DBCONTERM=</b>        | none                                           | •                |

| Option                    | Default Value                                | Valid in CONNECT |
|---------------------------|----------------------------------------------|------------------|
| DBCREATE_TABLE_OPTS=      | none                                         |                  |
| DBGEN_NAME=               | DBMS                                         | •                |
| DBINDEX=                  | NO                                           |                  |
| DBLIBINIT=                | none                                         |                  |
| DBLIBTERM=                | none                                         |                  |
| DBMAX_TEXT=               | 1024                                         | •                |
| DBMSTEMP=                 | NO                                           |                  |
| DBNULLKEYS=               | YES                                          |                  |
| DBPROMPT=                 | NO                                           | •                |
| DBSASLABEL=               | COMPAT                                       |                  |
| DEFER=                    | NO                                           | •                |
| DIRECT_EXE=               | none                                         |                  |
| DIRECT_SQL=               | YES                                          |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                           |                  |
| INSERT_SQL=               | YES                                          |                  |
| INSERTBUFF=               | automatically calculated based on row length |                  |
| LOGIN_TIMEOUT=            | 0                                            | •                |
| MULTI_DATASRC_OPT=        | NONE                                         |                  |
| POST_STMT_OPTS=           | none                                         |                  |
| PRESERVE_COL_NAMES=       | NO<br>See “Naming Conventions for Netezza”.  |                  |
| PRESERVE_TAB_NAMES=       | NO                                           |                  |
| PRESERVE_USER=            | NO                                           | •                |

| Option              | Default Value                                | Valid in CONNECT |
|---------------------|----------------------------------------------|------------------|
| QUALIFIER=          | none                                         |                  |
| QUERY_TIMEOUT=      | 0                                            | •                |
| QUOTE_CHAR=         | none                                         |                  |
| READBUFF=           | automatically calculated based on row length | •                |
| REREAD_EXPOSURE=    | NO                                           |                  |
| SCHEMA=             | the current connection schema                |                  |
| SPOOL=              | YES                                          |                  |
| SQL_FUNCTIONS=      | none                                         |                  |
| SQL_FUNCTIONS_COPY= | none                                         |                  |
| SQLGENERATION=      | none                                         |                  |
| STRINGDATES=        | NO                                           | •                |
| SUB_CHAR=           | none                                         |                  |
| SYNONYMS=           | YES                                          |                  |
| TRACE=              | NO                                           | •                |
| TRACEFILE=          | none                                         | •                |
| UTILCONN_TRANSIENT= | NO                                           |                  |

## Netezza LIBNAME Statement Examples

In this example, SERVER=, DATABASE=, USER=, and PASSWORD= are connection options.

```
libname mydblib netezza server=mysrv11 database=test
      user=myuser password=mypwd;

proc print data=mydblib.customers;
  where state='CA';
run;
```

In the next example, DATASRC=, USER=, and PASSWORD= are connection options. The NZSQL data source is configured in the ODBC Administrator Control Panel on Windows platforms or in the odbc.ini file or a similarly named configuration file on UNIX platforms.

```
libname mydblib netezza datasrc=NZSQL
      user=myuser password=mypwd;

proc print data=mydblib.customers;
      where state='CA';
run;
```

## Data Set Options for Netezza

SAS/ACCESS data set options in this table are supported for Netezza. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

*Table 29.2 SAS/ACCESS Data Set Options for Netezza*

| Option               | Default Value                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------|
| BL_DATAFILE=         | When BL_USE_PIPE=NO, creates a file in the current directory or with the default file specifications. |
| BL_DEFAULT_DIR=      | temporary file directory that is specified by the UTILLOC= system option                              |
| BL_DELETE_DATAFILE=  | YES (only when BL_USE_PIPE=NO)                                                                        |
| BL_DELIMITER=        | (the pipe symbol)                                                                                     |
| BL_OPTIONS=          | none                                                                                                  |
| BL_USE_PIPE=         | YES                                                                                                   |
| BULKLOAD=            | NO                                                                                                    |
| BULKUNLOAD=          | LIBNAME option value                                                                                  |
| DBCOMMIT=            | LIBNAME option value                                                                                  |
| DBCONDITION=         | none                                                                                                  |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                                                                                  |
| DBFORCE=             | NO                                                                                                    |

| Option                    | Default Value                |
|---------------------------|------------------------------|
| DBGEN_NAME=               | DBMS                         |
| DBINDEX=                  | LIBNAME option value         |
| DBKEY=                    | none                         |
| DBLABEL=                  | NO                           |
| DBLARGETABLE=             | none                         |
| DBMAX_TEXT=               | 1024                         |
| DBNULL=                   | YES                          |
| DBNULLKEYS=               | LIBNAME option value         |
| DBPROMPT=                 | LIBNAME option value         |
| DBSASTYPE=                | see “Data Types for Netezza” |
| DBTYPE=                   | see “Data Types for Netezza” |
| DISTRIBUTE_ON=            | none                         |
| ERRLIMIT=                 | 1                            |
| IGNORE_READ_ONLY_COLUMNS= | NO                           |
| INSERT_SQL=               | LIBNAME option value         |
| INSERTBUFF=               | LIBNAME option value         |
| NULLCHAR=                 | SAS                          |
| NULLCHARVAL=              | a blank character            |
| POST_STMT_OPTS=           | none                         |
| POST_TABLE_OPTS=          | none                         |
| PRE_STMT_OPTS=            | none                         |
| PRE_TABLE_OPTS=           | none                         |
| PRESERVE_COL_NAMES=       | LIBNAME option value         |
| QUALIFIER=                | LIBNAME option value         |

| Option         | Default Value        |
|----------------|----------------------|
| QUERY_TIMEOUT= | LIBNAME option value |
| READBUFF=      | LIBNAME option value |
| SASDATEFMT=    | none                 |
| SCHEMA=        | LIBNAME option value |
| SUB_CHAR=      | none                 |

---

## SQL Pass-Through Facility Specifics for Netezza

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page 690.

Here are the SQL pass-through facility specifics for the Netezza interface.

- The *dbms-name* is NETEZZA.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Netezza. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default *netezza* alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection-options](#). For additional LIBNAME options that are valid in the CONNECT statement, see [Table 29.1 on page 1052](#).

---

### CONNECT Statement Example

This example uses the DBCON alias to connection to the *mysrv1* Netezza Performance Server and execute a query. The connection alias is optional.

```
proc sql;
  connect to netezza as dbcon
  (server=mysrv1 database=test user=myuser password=mypwd);
```

```

select * from connection to dbcon
  (select * from customers where customer like '1%') ;
quit;

```

---

## Special Catalog Queries

SAS/ACCESS Interface to Netezza supports the following special queries. You can use the queries to call the ODBC-style catalog function application programming interfaces (APIs). Here is the general format of the special queries:

**Netezza::SQLAPI "parameter 1","parameter n"**

**Netezza::**

is required to distinguish special queries from regular queries.

**SQLAPI**

is the specific API that is being called. Neither Netezza:: nor SQLAPI are case sensitive.

**"parameter n"**

is a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, this call to SQLTables usually matches table names such as mytest and my\_test:

```
select * from connection to netezza (NETEZZA::SQLTables "test","","my_test");
```

Use the escape character to search only for the my\_test table:

```
select * from connection to netezza (NETEZZA::SQLTables "test","","my\_test");
```

SAS/ACCESS Interface to Netezza supports these special queries:

**Netezza::SQLTables <"Catalog", "Schema", "Table-name", "Type">**

returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.

**Netezza::SQLColumns <"Catalog", "Schema", "Table-name", "Column-name">**

returns a list of all columns that match the specified arguments. If you do not specify any argument, all accessible column names and information are returned.

**Netezza::SQLPrimaryKeys <"Catalog", "Schema", "Table-name">**

returns a list of all columns that compose the primary key that matches the specified table. A primary key can consist of one or more columns. If you do not specify any table name, this special query fails.

**Netezza::SQLSpecialColumns <"Identifier-type", "Catalog-name", "Schema-name", "Table-name", "Scope", "Nullable">**

returns a list of the optimal set of columns that uniquely identify a row in the specified table.

**Netezza::SQLStatistics <"Catalog", "Schema", "Table-name">**

returns a list of the statistics for the specified table name. You can specify the SQL\_INDEX\_ALL and SQL\_ENSURE options in the SQLStatistics API call. If you do not specify any table name argument, this special query fails.

**Netezza::SQLGetTypeInfo**

returns information about the data types that the Netezza Performance Server supports.

## Temporary Table Support for Netezza

SAS/ACCESS Interface to Netezza supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## Passing SAS Functions to Netezza

SAS/ACCESS Interface to Netezza passes the following SAS functions to Netezza for processing. Where the Netezza function name differs from the SAS function name, the Netezza name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                      |                                        |
|----------------------|----------------------------------------|
| ABS                  | LOG10 (log)                            |
| ARCOS (ACOS)         | LOWCASE (lower)                        |
| ARSIN (ASIN)         | MAX                                    |
| ATAN                 | MIN                                    |
| ATAN2                | MINUTE (date_part)                     |
| AVG                  | MOD (see note)                         |
| BAND (int4and)       | MONTH (date_part)                      |
| BNOT (int4not)       | QTR (date_part)                        |
| BLSHIFT (int4shl)    | REPEAT                                 |
| BRSHIFT (int4shr)    | SECOND (date_part)                     |
| BOR (int4or)         | SIGN                                   |
| BXOR (int4xor)       | SIN                                    |
| BYTE (chr)           | SQRT                                   |
| CEIL                 | STRIP (btrim)                          |
| COALESCE             | SUBSTR                                 |
| COMPRESS (translate) | SUM                                    |
| COS                  | TAN                                    |
| COUNT                | TRANSLATE (see note)                   |
| DAY (date_part)      | TRANWRD (translate)                    |
| EXP                  | TRIMN (rtrim)                          |
| FLOOR                | UPCASE (upper)                         |
| HOUR (date_part)     | WEEK[<SAS date value>, V'] (date_part) |
| INDEX (position)     | WEEKDAY (date_part)                    |
| LOG (ln)             | YEAR (date_part)                       |

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

**Note:** If more than three arguments are provided for the TRANSLATE function, then TRANSLATE is performed by SAS and the function is not passed to the database.

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to Netezza. Due to incompatibility in date and time functions between Netezza and SAS, Netezza might not process them correctly. Check your results to determine whether these functions are working as expected.

|                     |                                         |
|---------------------|-----------------------------------------|
| DATE (current_date) | TIMEPART (cast)                         |
| DATEPART (cast)     | TODAY (current_date)                    |
| DATETIME (now)      | WEEK[<SAS date value>] (date_part)      |
| LENGTH              | WEEK[<SAS date value>, 'U'] (date_part) |
| ROUND               | WEEK[<SAS date value>, 'W'] (date_part) |
| TIME (current_time) | WEEKDAY (date_part)                     |

## Passing Joins to Netezza

For a multiple libref join to pass to Netezza, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- port (PORT=)
- data source (DATASRC=, if specified)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

# Bulk Loading and Unloading for Netezza

---

## Loading

---

### Overview of Bulk Loading for Netezza

Bulk loading is the fastest way to insert large numbers of rows into a Netezza table. You must specify **BULKLOAD=YES** to use the bulk-load facility. The bulk-load facility uses the Netezza Remote External Table interface to move data from the client to the Netezza Performance Server.

---

### Data Set Options for Bulk Loading

Here are the Netezza bulk-load data set options. For detailed information about these options, see [Data Set Options for Relational Databases on page 370](#).

- **BL\_DATAFILE=**
- **BL\_DELETE\_DATAFILE=**
- **BL\_DELIMITER=**
- **BL\_OPTIONS=**
- **BL\_USE\_PIPE=**
- **BULKLOAD=**

---

### Bulk Loading Examples for Netezza

This first example shows how you can use a SAS data set, SASFLT.FLT98, to create and load a large Netezza table, FLIGHTS98:

```
libname sasflt 'SAS-library';
libname net_air netezza user=myuser pwd=mypwd
      server=air2 database=flights;

proc sql;
  create table net_air.flights98
    (bulkload=YES bl_options="logdir 'c:\temp\netlogs\'")
     as select * from sasflt.flt98;
quit;
```

You can use BL\_OPTIONS= to pass specific Netezza options to the bulk-loading process. The LOGDIR option specifies the directory for the Nzbad and Nzlog files to be generated during the load.

This next example shows how you can append the SAS data set, SASFLT.FLT98, to the existing Netezza table, ALLFLIGHTS. The BL\_USE\_PIPE=NO option forces SAS/ACCESS Interface to Netezza to write data to a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the load has completed.

```
proc append base=net_air.allflights
  (BULKLOAD=YES
   BL_DATAFILE='/tmp/fltdata.dat'
   BL_USE_PIPE=NO
   BL_DELETE_DATAFILE=NO)
  data=sasflt.flt98;
  run;
```

## Unloading

### Overview of Bulk Unloading for Netezza

Bulk unloading is the fastest way to read large numbers of rows from a Netezza table. To use the bulk-unloading facility, specify [BULKUNLOAD=](#) on page 493 YES. The bulk-unloading facility uses the Netezza Remote External Table interface to move data from the client to the Netezza Performance Server into SAS.

### Data Set Options for Bulk Unloading

Here are the Netezza bulk-unloading data set options.

- [BL\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DELIMITER=](#)
- [BL\\_USE\\_PIPE=](#)
- [BULKUNLOAD=](#)

### Bulk Unloading Examples for Netezza

This first example shows how you can read the large Netezza table, FLIGHTS98, to create and populate a SAS data set, SASFLT.FLT98:

```
libname sasflt 'SAS-library';
libname net_air netezza user=myuser pwd=mypwd
```

```

server=air2 database=flights;

proc sql;
create table sasflt.flt98
      as select * from net_air.flights98
      (bulkunload=YES bl_options="logdir 'c:\temp\netlogs'" );
quit;

```

You can use BL\_OPTIONS= to pass specific Netezza options to the bulk unloading process. The LOGDIR option specifies the directory for the Nzbad and Nzlog files to be generated during the bulk unloading.

This next example shows how you can append the contents of the Netezza table, ALLFLIGHTS, to an existing SAS data set, SASFLT.FLT98. The BL\_USE\_PIPE=NO option forces SAS/ACCESS Interface to Netezza to read data from a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the bulk unloading has completed.

```

proc append base=sasflt.flt98
  data=net_air.allflights
  (BULKUNLOAD=YES
   BL_DATAFILE='/tmp/fltdata.dat'
   BL_USE_PIPE=NO
   BL_DELETE_DATAFILE=NO);
run;

```

## Naming Conventions for Netezza

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Netezza interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options determine how SAS/ACCESS Interface to Netezza handles case sensitivity. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Netezza is not case sensitive, and all names default to lowercase.

Netezza objects include tables, views, and columns. Follow these naming conventions:

- A name must be from 1 to 128 characters long.
- A name must begin with a letter (A through Z), diacritic marks, or non-Latin characters (200-377 octal).
- A name cannot begin with an underscore (\_). Leading underscores are reserved for system objects.

- Names are not case sensitive. For example, CUSTOMER and Customer are the same, but object names are converted to lowercase when they are stored in the Netezza database. However, if you enclose a name in quotation marks, it is case sensitive.
- A name cannot be a Netezza reserved word, such as WHERE or VIEW.
- A name cannot be the same as another Netezza object that has the same type.

For more information, see your *Netezza Database User's Guide*.

## Data Types for Netezza

### Overview

Every column in a table has a name and a data type. The data type tells Netezza how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Netezza data types, null and default values, and data conversions.

For more information about Netezza data types and to determine which data types are available for your version of Netezza, see your *Netezza Database User's Guide*.

SAS/ACCESS Interface to Netezza does not directly support TIMETZ or INTERVAL types. Any columns using these types are read into SAS as character strings.

## Supported Netezza Data Types

Here are the data types that are supported by Netezza:

- Character data:

|                   |                      |
|-------------------|----------------------|
| CHAR( <i>n</i> )  | NVARCHAR( <i>n</i> ) |
| NCHAR( <i>n</i> ) | VARCHAR( <i>n</i> )  |

- Numeric data:

|                               |             |
|-------------------------------|-------------|
| BIGINT                        | REAL        |
| BYTEINT                       | SMALLINT    |
| DECIMAL   DEC   NUMERIC   NUM | ST_Geometry |
| DOUBLE   DOUBLE PRECISION     | VARBINARY   |
| INTEGER                       |             |

**Note:** Support for ST\_Geometry and VARBINARY was added in SAS 9.4M2.

You might observe errors in the SAS log when you load large numeric values with more than 15 digits of precision. For more information, see “[Loading Large Numeric Values](#)” on page 1065.

Columns that use the binary data types, such as ST\_GEOMETRY and VARBINARY, do not support some of the common query processing operations. For example, binary data type columns cannot be used in ordering, grouping, or in magnitude comparisons. They cannot be used in aggregates such as sum, avg, distinct, min, or max comparisons. The binary data cannot be implicitly or explicitly cast to other data types.

- Date and time data:

DATE

TIME

DATETIME

SQL date and time data types are collectively called datetime values. The SQL data types for dates, times, and timestamps are listed here. Be aware that columns of these data types can contain data values that are out of range for SAS.

## Loading Large Numeric Values

Be aware that when performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that are read are rounded to meet this condition. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see [“Your Options When Choosing Your Needed Degree of Precision” on page 10](#).

By default, when SAS/ACCESS loads numeric data with more than 15 digits of precision, errors are written to the log and the data fails to load. You can choose to load the data and suppress the errors by setting the TRUNCATE\_BIGINT environment variable to YES before you start SAS.

## Default SAS Formats for Netezza NUMERIC Data Types

This table shows the correlation between the Netezza NUMERIC data types and the default SAS formats that are created from that data type.

**Table 29.3 Default SAS Formats for Netezza NUMERIC Data Types**

| Netezza NUMERIC Data Type | Rules                      | Default SAS Format  |
|---------------------------|----------------------------|---------------------|
| NUMERIC( $p$ )            | $0 < p \leq 32$            | ( $p + 1$ ).0       |
| NUMERIC( $p,s$ )          | $p > 0, s < 0,  s  < p$    | ( $p +  s  + 1$ ).0 |
| NUMERIC( $p,s$ )          | $p > 0, s < 0,  s  \geq p$ | ( $p +  s  + 1$ ).0 |

| Netezza NUMERIC Data Type | Rules                    | Default SAS Format |
|---------------------------|--------------------------|--------------------|
| NUMERIC( $p,s$ )          | $p > 0, s > 0, s < p$    | ( $p + 2$ ). $s$   |
| NUMERIC( $p,s$ )          | $p > 0, s > 0, s \geq p$ | ( $s + 3$ ). $s$   |
| NUMERIC( $p$ )            | $p > 32$                 | BEST22.            |
| NUMERIC                   | $p, s$ unspecified       | BEST22.            |

The general form of a Netezza number is NUMERIC( $p,s$ ) where  $p$  is the precision and  $s$  is the scale of the number. Netezza specifies precision as the total number of digits, with a valid range of –84 to 127. However, a negative scale means that the number is rounded to the specified number of places to the left of the decimal. For example, if the number 1,234.56 is specified as data type NUMERIC(8,–2), it is rounded to the nearest hundred and stored as 1,200.

## Netezza Null Values

Netezza has a special value called NULL. A Netezza NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Netezza NULL value, it interprets it as a SAS missing value.

You can specify a column in a Netezza table so that it requires data. To do this in SQL, you specify a column as NOT NULL. This tells SQL to allow only a row to be added to a table if a value exists for the field. For example, NOT NULL assigned to the CUSTOMER field in the SASDEMO.CUSTOMER table does not allow a row to be added unless there is a value for CUSTOMER. When creating a Netezza table with SAS/ACCESS, you can use the [DBNULL=](#) data set option to indicate whether NULL is a valid value for specified columns.

You can also specify Netezza columns as NOT NULL DEFAULT. For more information about using the NOT NULL DEFAULT value, see your *Netezza Database User's Guide*.

Once you know whether a Netezza column enables NULLs or the host system provides a default value for a column that is specified as NOT NULL WITH DEFAULT, you can write selection criteria and enter values to update a table. Unless a column is specified as NOT NULL or NOT NULL DEFAULT, it allows NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how the DBMS handles SAS missing character values, use the [NULLCHAR=](#) and [NULLCHARVAL=](#) data set options.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Netezza assigns to SAS variables when using the [LIBNAME statement](#) to read from a Netezza table. These default formats are based on Netezza column attributes.

**Table 29.4 LIBNAME Statement: Default SAS Formats for Netezza Data Types**

| Netezza Data Type                  | SAS Data Type | Default SAS Format |
|------------------------------------|---------------|--------------------|
| BIGINT                             | numeric       | 20.                |
| CHAR( <i>n</i> ) <sup>1</sup>      | character     | \$ <i>w</i> .      |
| DATE                               | numeric       | DATE9.             |
| DECIMAL( <i>p,s</i> ) <sup>2</sup> | numeric       | <i>w.d</i>         |
| DOUBLE                             | numeric       | none               |
| INTEGER                            | numeric       | 11.                |
| NCHAR( <i>n</i> ) <sup>1</sup>     | character     | \$ <i>w</i> .      |
| NUMERIC( <i>p,s</i> ) <sup>2</sup> | numeric       | <i>w.d</i>         |
| NVARCHAR( <i>n</i> ) <sup>1</sup>  | character     | \$ <i>w</i> .      |
| REAL                               | numeric       | none               |
| SMALLINT                           | numeric       | 6.                 |
| BYTEINT                            | numeric       | 4.                 |
| ST_GeOMETRY                        | character     | \$HEX2 <i>w</i> .  |
| TIME                               | numeric       | TIME8.             |
| TIMESTAMP                          | numeric       | DATETIME25.6       |
| VARBINARY                          | character     | \$HEX2 <i>w</i> .  |
| VARCHAR( <i>n</i> ) <sup>1</sup>   | character     | \$ <i>w</i> .      |
| VINBINARY                          | numeric       | 8.                 |

<sup>1</sup> *n* in Netezza character data types is equivalent to *w* in SAS formats.

<sup>2</sup> *p* and *s* in Netezza numeric data types are equivalent to *w* and *d* in SAS formats.

This table shows the default Netezza data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 29.5 LIBNAME Statement: Default Netezza Data Types for SAS Variable Formats**

| SAS Variable Format | Netezza Data Type                  |
|---------------------|------------------------------------|
| <i>w.d</i>          | DECIMAL( <i>p,s</i> ) <sup>2</sup> |

| SAS Variable Format | Netezza Data Type                |
|---------------------|----------------------------------|
| other numerics      | DOUBLE                           |
| \$w.                | VARCHAR( <i>n</i> ) <sup>1</sup> |
| datetime formats    | TIMESTAMP                        |
| date formats        | DATE                             |
| time formats        | TIME                             |

<sup>1</sup> *n* in Netezza character data types is equivalent to *w* in SAS formats.

<sup>2</sup> *p* and *s* in Netezza numeric data types are equivalent to *w* and *d* in SAS formats.

---

## Sample Programs for Netezza

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to ODBC

|                                                                                |      |
|--------------------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to ODBC</i> .....              | 1070 |
| <i>Introduction to SAS/ACCESS Interface to ODBC</i> .....                      | 1070 |
| Overview .....                                                                 | 1070 |
| ODBC Concepts .....                                                            | 1071 |
| <i>LIBNAME Statement for the ODBC Engine</i> .....                             | 1075 |
| Overview .....                                                                 | 1075 |
| Arguments .....                                                                | 1075 |
| ODBC LIBNAME Statement Examples .....                                          | 1080 |
| <i>Data Set Options for ODBC</i> .....                                         | 1081 |
| <i>SQL Pass-Through Facility Specifics for ODBC</i> .....                      | 1083 |
| Key Information .....                                                          | 1083 |
| CONNECT Statement Examples .....                                               | 1084 |
| Connection to Component Examples .....                                         | 1084 |
| Special Catalog Queries .....                                                  | 1086 |
| <i>Autopartitioning Scheme for ODBC</i> .....                                  | 1087 |
| Overview .....                                                                 | 1087 |
| Autopartitioning Restrictions .....                                            | 1088 |
| Nullable Columns .....                                                         | 1088 |
| Using WHERE Clauses .....                                                      | 1088 |
| Using DBSLICEPARAM= .....                                                      | 1089 |
| Using DBSLICE= .....                                                           | 1089 |
| Configuring Microsoft SQL Server Partitioned Views for Use with DBSLICE= ..... | 1089 |
| <i>DBLOAD Procedure Specifics for ODBC</i> .....                               | 1092 |
| Overview .....                                                                 | 1092 |
| Examples .....                                                                 | 1093 |
| <i>Temporary Table Support for ODBC</i> .....                                  | 1094 |
| <i>Passing SAS Functions to ODBC</i> .....                                     | 1094 |
| <i>Passing Joins to ODBC</i> .....                                             | 1095 |
| <i>Bulk Loading for ODBC</i> .....                                             | 1096 |
| <i>Locking in the ODBC Interface</i> .....                                     | 1096 |
| <i>Naming Conventions for ODBC</i> .....                                       | 1098 |

|                                    |             |
|------------------------------------|-------------|
| <b>Data Types for ODBC</b>         | <b>1098</b> |
| Overview                           | 1098        |
| ODBC Null Values                   | 1099        |
| LIBNAME Statement Data Conversions | 1099        |
| <b>Sample Programs for ODBC</b>    | <b>1101</b> |

---

# System Requirements for SAS/ACCESS Interface to ODBC

You can find information about system requirements for SAS/ACCESS Interface to ODBC in the following locations:

- [System Requirements for SAS/ACCESS Interface to ODBC with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

# Introduction to SAS/ACCESS Interface to ODBC

## Overview

For available SAS/ACCESS features, see [ODBC supported features on page 110](#). For more information about ODBC, see your ODBC documentation.

Beginning in SAS Viya 3.3, SAS/ACCESS Interface to ODBC includes SAS Data Connector to ODBC. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- “[Where to Specify Data Connector Options](#)” in *SAS Cloud Analytic Services: User’s Guide*
- “[ODBC Data Connector](#)” in *SAS Cloud Analytic Services: User’s Guide*

# ODBC Concepts

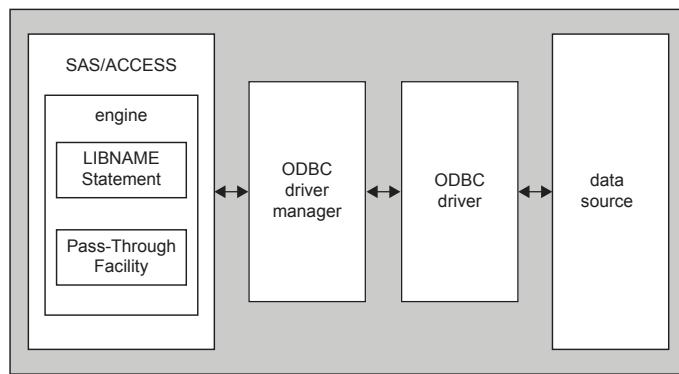
## Overview

Open database connectivity (ODBC) standards provide a common interface to a variety of data sources, including dBASE, Microsoft Access, Microsoft SQL Server, Oracle, and Paradox. The goal of ODBC is to enable access to data from any application, regardless of which DBMS handles the data. ODBC accomplishes this by inserting a middle layer—consisting of an ODBC driver manager and an ODBC driver—between an application and the target DBMS. The purpose of this layer is to translate application data queries into commands that the DBMS understands. Specifically, ODBC standards specify application programming interfaces (APIs) that enable applications such as SAS software to access a database. For all of this to work, both the application and the DBMS must be ODBC-compliant. This means that the application must be able to issue ODBC commands, and the DBMS must be able to respond to these.

Here are the basic components and features of ODBC.

Three components provide ODBC functionality: the client interface, the ODBC driver manager, and the ODBC driver for the data source with which you want to work, as shown below.

*Figure 30.1 The ODBC Interface to SAS*



For PC and UNIX environments, SAS provides SAS/ACCESS Interface to ODBC as the client interface. Consisting of the ODBC driver manager and the ODBC driver, the client setup with which SAS/ACCESS Interface to ODBC works is quite different between the two platforms.

## ODBC on a PC Platform

On the PC side, the Microsoft ODBC Data Source Administrator is the ODBC driver manager. You can open the ODBC Data Source Administrator from the Windows control panel. Through a series of dialog boxes, you can create an ODBC data

source name (DSN). You can select a particular ODBC driver for the database with which you want to work from the list of available drivers. You can then provide specific connection information for the database that the specific driver can access.

**USER DSN**

specific to an individual user. It is available only to the user who creates it.

**SYSTEM DSN**

not specific to an individual user. Anyone with permission to access the data source can use it.

**FILE DSN**

not specific to an individual user. It can be shared among users even though it is created locally. Because this DSN is file-based, it contains all information that is required to connect to a data source.

You can create multiple DSNs in this way and then reference them in your PC-based SAS/ACCESS Interface to ODBC code.

When you use the ODBC Data Source Administrator on the PC to create your ODBC data sources, the ODBC drivers for the particular databases that you want to access are often in the list of available drivers. This is especially true for drivers for the more common databases. If the ODBC driver that you want is not listed, you must work to obtain one.

---

## ODBC on a UNIX Platform

ODBC on UNIX works a bit differently. The ODBC driver manager and ODBC drivers on the PC are available by default, so you need only plug them in. Because these components are not generally available on UNIX, you must instead work with third-party vendors to obtain them.

When you submit SAS/ACCESS Interface to ODBC code, SAS looks first for an ODBC driver manager. It checks the directories that are listed in such environment variables as LD\_LIBRARY\_PATH, LIBPATH, or SHLIB\_PATH, depending on your UNIX platform. It uses the first ODBC driver manager that it finds.

The ODBC driver manager then checksINI files—either a stand-alone ODBC.INI file, or a combination of ODBC.INI and ODBCINST.INI files—for the DSNs that you specified in your code. To make sure that the intended INI files are referenced, you can use such environment variables as ODBCINI or ODBCSYSINI, depending on how your INI files are set up. You can set up global INI files for all your users, or you can set up INI files for single users or groups of users. This is similar to using the ODBC Data Source Administrator to create either SYSTEM or USER DSNs for PC platforms. One or more INI files include a section for each DSN, and each section includes specific connection information for each data source from which you want to enable access to data. Some ODBC driver vendors provide tools with which you can build one or more of your INI files. However, editing a sample generic INI file that is provided with the ODBC driver is often done manually.

Most database vendors (such as SAP, Oracle, or DB2) include ODBC drivers for UNIX platforms. To use SAS/ACCESS Interface to ODBC, pair an ODBC driver manager that is based in UNIX with your ODBC driver that is also based in UNIX. Freeware ODBC driver managers for UNIX such as unixODBC are generally available for download. Another alternative is to obtain the required ODBC client components for UNIX platforms from third-party vendors who market both ODBC drivers for databases and an ODBC driver manager that works with these drivers.

To use SAS/ACCESS Interface to ODBC, you can select any ODBC client solution that you want as long as it is ODBC-compliant.

## ODBC for PC and UNIX Platforms

These concepts are common across both PC and UNIX platforms.

- ODBC uses SQL syntax for queries and statement execution, or for statements that are executed as commands. However, all databases that support ODBC are not necessarily SQL databases. For example, many databases do not have system tables. Also, the term *table* can describe a variety of items—including a file, a part of a file, a group of files, a typical SQL table, generated data, or any potential source of data. This is an important distinction. All ODBC data sources respond to a base set of SQL statements such as SELECT, INSERT, UPDATE, DELETE, CREATE, and DROP in their simplest forms. However, some databases do not support other statements and more complex forms of SQL statements.
- The ODBC standard allows for various levels of conformance that is generally categorized as low, medium, and high. As previously mentioned, the level of SQL syntax that is supported varies. Also, some driver might not support many programming interfaces. SAS/ACCESS Interface to ODBC works with API calls that conform to the lowest level of ODBC compliance, Level 1. However, it does use some Level 2 API calls if they are available.

SAS programmers or end users must make sure that their particular ODBC driver supports the SQL syntax to be used. If the driver supports a higher level of API conformance, some advanced features are available through the PROC SQL CONNECT statement and special queries that SAS/ACCESS Interface to ODBC supports. For more information, see “[Special Catalog Queries](#)” on page [1086](#).

- The ODBC manager and drivers return standard operation states and custom text for any warnings or errors. The state variables and their associated text are available through the SAS SYSDBRC and SYSDBMSG macro variables.

## Key Considerations When Using ODBC Drivers and SAS on UNIX

SAS/ACCESS Interface to ODBC on UNIX allows SAS customers to present data from a wide variety of external data sources. Many customers using SAS on UNIX have had success using SAS/ACCESS Interface to ODBC with their ODBC client setups. These setups consist of an ODBC driver manager and ODBC drivers for the specific data sources to which customers need access. Critical to this success are the quality and completeness of third-party ODBC client components on UNIX that customers have chosen to use.

To maximize your chances of success, your ODBC driver must comply with the ODBC 3.5 (or later) specification. It must also support the call sequences that SAS/ACCESS Interface to ODBC sends to the driver. Specifically, your ODBC driver manager and ODBC driver must support these ODBC calls:

|                     |                     |
|---------------------|---------------------|
| SQLAllocConnect     | SQLFreeStmt         |
| SQLAllocEnv         | SQLGetConnectAttr   |
| SQLAllocHandle      | SQLGetConnectOption |
| SQLAllocStmt        | SQLGetCursorName    |
| SQLBindCol          | SQLGetDiagRec       |
| SQLBindParameter    | SQLGetFunctions     |
| SQLBulkOperations   | SQLGetInfo          |
| SQLCancel           | SQLGetStmtAttr      |
| SQLColAttribute     | SQLGetStmtOption    |
| SQLColumnPrivileges | SQLGetTypeInfo      |
| SQLColumns          | SQLMoreResults      |
| SQLConnect          | SQLNumResultCols    |
| SQLDataSources      | SQLPrepare          |
| SQLDescribeCol      | SQLPrepareW         |
| SQLDescribeColW     | SQLPrimaryKeys      |
| SQLDisconnect       | SQLProcedureColumns |
| SQLDriverConnect    | SQLProcedures       |
| SQLEndTran          | SQLRowCount         |
| SQLExecDirect       | SQLSetConnectAttr   |
| SQLExecDirectW      | SQLSetConnectOption |
| SQLExecute          | SQLSetEnvAttr       |
| SQLExtendedFetch    | SQLSetPos           |
| SQLFetch            | SQLSetStmtAttr      |
| SQLFetchScroll      | SQLSetStmtOption    |
| SQLForeignKeys      | SQLSpecialColumns   |
| SQLFreeConnect      | SQLStatistics       |
| SQLFreeEnv          | SQLTablePrivileges  |
| SQLFreeHandle       | SQLTables           |

SAS/ACCESS Interface to ODBC sends a sequence of ODBC calls to the ODBC driver that you have chosen. The types and sequence in which these calls are made are compliant with the ODBC specification. If your ODBC driver fails to return the correct result or fails to work with SAS/ACCESS Interface to ODBC, here are your options.

- Make sure that you are running the current versions of your ODBC client components.
- Try to connect using a query tool that is not SAS. Most third-party ODBC driver and driver manager sources include such a query tool with their offerings. However, keep in mind that in some cases you might be able to connect using a query tool (that is not SAS) but not with SAS/ACCESS Interface to ODBC. This is because SAS calls might make a wider range of ODBC calls than an ODBC query tool that is not SAS would make.
- SAS Technical Support offers additional tools that can help you identify the root of your ODBC-related client problems and subsequently debug them.
- You can address some ODBC client issues by using certain SAS/ACCESS Interface to ODBC options or alternative engines to SAS/ACCESS Interface to ODBC.

- Once you have determined that your ODBC client issues are not related to SAS, you need to report your debugging results to your ODBC client providers. If you received your ODBC client components from a commercial ODBC driver vendor, you can work through that vendor's technical support. If you use freeware or open-source ODBC client components—where formal technical support is not always available—your only recourse might be to communicate with the freeware user community.

SAS has not validated all ODBC drivers on the market and therefore makes no claims of certification or support.

# LIBNAME Statement for the ODBC Engine

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to ODBC supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing ODBC.

**LIBNAME** *libref odbc <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *odbc*

specifies the SAS/ACCESS engine name for the ODBC interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to ODBC in many different ways. Specify only one of these methods for each connection because they are mutually exclusive.

- USER=, PASSWORD=, DATASRC=
- COMPLETE=
- NOPROMPT=
- PROMPT=

■ REQUIRED=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**USER=<'>ODBC-user-name<'>**

lets you connect to an ODBC database with a user ID that is different from the default ID. USER= is optional.

Alias: UID=

**PASSWORD=<'>ODBC-password<'>**

specifies the ODBC password that is associated with your user ID.

PASSWORD= is optional. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. If you do not want to enter your ODBC password in uncoded text on this statement, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

Alias: PWD=

**DATASRC=<'>ODBC-data-source<'>**

specifies the ODBC data source to which you want to connect. For Windows platforms, data sources must be configured by using the ODBC Driver Manager. For UNIX platforms, data sources must be configured by modifying the .odbc.ini file. This option indicates that the connection is attempted using the ODBC SQLConnect API, which requires a data source name. You can also use a user ID and password with DSN=. If you want to use an ODBC file called DSN, instead of specifying DATASRC=<'>ODBC-data-source<'>, use the PROMPT= or NOPROMPT= option, followed by "filedsn=(name-of-your-file-dsn);". Here is an example:

```
libname mydblib odbc noprompt="filedsn=d:\share\msafiledsn.dsn";
```

Alias: DATABASE=, DB=, DSN=

**COMPLETE=<'>ODBC-connection-options<'>**

specifies connection options for your data source or database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, you are prompted with a dialog box that displays the values from the COMPLETE= connection string. You can edit any field before you connect to the data source. This option is not supported on UNIX platforms. See your ODBC driver documentation for more details.

**NOPROMPT=<'>ODBC-connection-options<'>**

specifies connection options for your data source or database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned. No dialog box is displayed to help you complete the connection string.

**PROMPT=<'>ODBC-connection-information<'>**

specifies connection options for your data source or database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately try to connect to the DBMS. A dialog box is displayed in SAS windowing environment instead that contains the values that you

entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the data source.

Restriction: This option is not available on UNIX platforms.

**REQUIRED=<'>ODBC-connection-options<'>**

specifies connection options for your data source or database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, a dialog box prompts you for the connection options. REQUIRED= lets you modify only required fields in the dialog box.

Restriction: not supported on UNIX platforms

See your ODBC driver documentation for a list of the ODBC connection options that your ODBC driver supports.

The following ODBC connection options are not supported on UNIX:

- COMPLETE=
- PROMPT=
- REQUIRED=

**LIBNAME-options**

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to ODBC, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 30.1** SAS/ACCESS LIBNAME Options for ODBC

| Option      | Default Value                                                                            | Valid in CONNECT |
|-------------|------------------------------------------------------------------------------------------|------------------|
| ACCESS=     | none                                                                                     |                  |
| AUTHDOMAIN= | none                                                                                     |                  |
| AUTOCOMMIT= | data-source specific                                                                     | •                |
| BL_LOG=     | none                                                                                     |                  |
| BL_OPTIONS= | none                                                                                     |                  |
| BULKLOAD=   | NO                                                                                       |                  |
| CONNECTION= | UNIQUE when data source supports only one cursor per connection,<br>SHAREDREAD otherwise | •                |

| Option               | Default Value                                  | Valid in CONNECT |
|----------------------|------------------------------------------------|------------------|
| CONNECTION_GROUP=    | none                                           | •                |
| CURSOR_TYPE=         | FORWARD_ONLY                                   | •                |
| DATETIME2=           | NO                                             |                  |
| DBCOMMIT=            | 1000 when inserting rows, 0 when updating rows |                  |
| DBCONINIT=           | none                                           | •                |
| DBCONTERM=           | none                                           | •                |
| DBCREATE_TABLE_OPTS= | none                                           |                  |
| DBGEN_NAME=          | DBMS                                           | •                |
| DBINDEX=             | NO                                             |                  |
| DBLIBINIT=           | none                                           |                  |
| DBLIBTERM=           | none                                           |                  |
| DBMAX_TEXT=          | 1024                                           | •                |
| DBMSTEMP=            | NO                                             |                  |
| DBNULLKEYS=          | YES                                            |                  |
| DBNULLWHERE=         | YES                                            |                  |
| DBPROMPT=            | NO                                             | •                |
| DBSASLABEL=          |                                                |                  |
| DBSLICEPARM=         | THREADED_APPS,2<br>or<br>THREADED_APPS,3       |                  |
| DEFER=               | NO                                             | •                |
| DELETE_MULT_ROWS=    | NO                                             |                  |
| DIRECT_EXE=          | none                                           |                  |
| DIRECT_SQL=          | YES                                            |                  |

| Option                    | Default Value                            | Valid in CONNECT |
|---------------------------|------------------------------------------|------------------|
| IGNORE_READ_ONLY_COLUMNS= | NO                                       |                  |
| INSERT_SQL=               | data-source specific                     |                  |
| INSERTBUFF=               | based on row length                      |                  |
| KEYSET_SIZE=              | 0                                        |                  |
| LOGIN_TIMEOUT=            | 0                                        |                  |
| MULTI_DATASRC_OPT=        | NONE                                     |                  |
| PRESERVE_COL_NAMES=       | see “Naming Conventions for ODBC”        |                  |
| PRESERVE_COMMENTS=        | NO                                       | •                |
| PRESERVE_GUID=            | YES                                      |                  |
| PRESERVE_TAB_NAMES=       | see “Naming Conventions for ODBC”        |                  |
| PRESERVE_USER=            | NO                                       |                  |
| QUALIFIER=                | none                                     |                  |
| QUERY_TIMEOUT=            | 0                                        | •                |
| QUOTE_CHAR=               | none                                     |                  |
| READ_ISOLATION_LEVEL=     | RC (see “Locking in the ODBC Interface”) | •                |
| READ_LOCK_TYPE=           | ROW                                      | •                |
| READBUFF=                 | 0                                        |                  |
| REREAD_EXPOSURE=          | NO                                       | •                |
| SCHEMA=                   | none                                     |                  |
| SPOOL=                    | YES                                      |                  |
| SQL_FUNCTIONS=            | none                                     |                  |
| SQL_FUNCTIONS_COPY=       | none                                     |                  |

| Option                  | Default Value                            | Valid in CONNECT |
|-------------------------|------------------------------------------|------------------|
| STRINGDATES=            | NO                                       | •                |
| TRACE=                  | NO                                       | •                |
| TRACEFILE=              | none                                     | •                |
| UPDATE_ISOLATION_LEVEL= | RC (see “Locking in the ODBC Interface”) |                  |
| UPDATE_LOCK_TYPE=       | ROW                                      | •                |
| UPDATE_MULT_ROWS=       | NO                                       |                  |
| UPDATE_SQL=             | driver-specific                          |                  |
| USE_ODBC_CL=            | NO                                       | •                |
| UTILCONN_TRANSIENT=     | NO                                       |                  |
| WARN_BIGINT=            | NO                                       |                  |

## ODBC LIBNAME Statement Examples

In this example, USER=, PASSWORD=, and DATASRC= are connection options.

```
libname mydblib odbc user=myusr1 password=mypwd1 datasrc=mydatasource;
```

In this next example, the libref MYLIB uses the ODBC engine to connect to an Oracle database. The connection options are USER=, PASSWORD=, and DATASRC=.

```
libname mydblib odbc datasrc=mydatasource user=myusr1 password=mypwd1;
```

```
proc print data=mydblib.customers;
  where state='CA';
run;
```

In the next example, the libref MYDBLIB uses the ODBC engine to connect to a Microsoft SQL Server database. The connection option is NOPROMPT=.

```
libname mydblib odbc
  noprompt="uid=myusr1;pwd=mypwd1;dsn=sqlservr;"
  stringdates=yes;
```

```
proc print data=mydblib.customers;
  where state='CA';
run;
```

# Data Set Options for ODBC

All SAS/ACCESS data set options in this table are supported for ODBC. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 30.2** SAS/ACCESS Data Set Options for ODBC

| Option               | Default Value        |
|----------------------|----------------------|
| BULKLOAD=            | LIBNAME option value |
| CURSOR_TYPE=         | LIBNAME option value |
| DATETIME2=           | NO                   |
| DBCOMMIT=            | LIBNAME option value |
| DBCONDITION=         | none                 |
| DBCREATE_TABLE_OPTS= | LIBNAME option value |
| DBFORCE=             | NO                   |
| DBGEN_NAME=          | DBMS                 |
| DBINDEX=             | LIBNAME option value |
| DBKEY=               | none                 |
| DBLABEL=             | NO                   |
| DBLARGETABLE=        | none                 |
| DBMAX_TEXT=          | 1024                 |
| DBNULL=              | YES                  |
| DBNULLKEYS=          | LIBNAME option value |
| DBNULLWHERE=         | LIBNAME option value |
| DBPROMPT=            | LIBNAME option value |
| DBSASLABEL=          | COMPAT               |

| Option                    | Default Value                         |
|---------------------------|---------------------------------------|
| DBSASTYPE=                | see "Data Types for ODBC"             |
| DBSLICE=                  | none                                  |
| DBSLICEPARM=              | THREADED_APPS,2 or<br>THREADED_APPS,3 |
| DBTYPE=                   | see "Data Types for ODBC"             |
| ERRLIMIT=                 | 1                                     |
| IGNORE_READ_ONLY_COLUMNS= | NO                                    |
| INSERT_SQL=               | LIBNAME option value                  |
| INSERTBUFF=               | LIBNAME option value                  |
| KEYSET_SIZE=              | LIBNAME option value                  |
| NULLCHAR=                 | SAS                                   |
| NULLCHARVAL=              | a blank character                     |
| PRESERVE_COL_NAMES=       | LIBNAME option value                  |
| QUALIFIER=                | LIBNAME option value                  |
| QUERY_TIMEOUT=            | LIBNAME option value                  |
| READ_ISOLATION_LEVEL=     | LIBNAME option value                  |
| READ_LOCK_TYPE=           | LIBNAME option value                  |
| READBUFF=                 | LIBNAME option value                  |
| SASDATEFMT=               | none                                  |
| SCHEMA=                   | LIBNAME option value                  |
| UPDATE_ISOLATION_LEVEL=   | LIBNAME option value                  |
| UPDATE_LOCK_TYPE=         | LIBNAME option value                  |
| UPDATE_SQL=               | LIBNAME option value                  |

---

# SQL Pass-Through Facility Specifics for ODBC

---

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the ODBC interface.

- The *dbms-name* is ODBC.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to ODBC. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default `odbc` alias is used. The functionality of multiple connections to the same ODBC data source might be limited by the particular data source driver.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection-options](#). Not all ODBC drivers support all of these arguments. See your driver documentation for more information.
- On some DBMSs, the *DBMS-SQL-query* argument can be a DBMS-specific SQL EXECUTE statement that executes a DBMS stored procedure. However, if the stored procedure contains more than one query, only the first query is processed.
- These options are available with the CONNECT statement.
  - [AUTOCOMMIT=](#)
  - [CURSOR\\_TYPE=](#)
  - [KEYSET\\_SIZE=](#)
  - [QUERY\\_TIMEOUT=](#)
  - [READBUFF=](#)
  - [READ\\_ISOLATION\\_LEVEL=](#)
  - [TRACE=](#)
  - [TRACEFILE=](#)
  - [USE\\_ODBC\\_CL=](#)
  - [UTILCONN\\_TRANSIENT=](#)

## CONNECT Statement Examples

These examples use ODBC to connect to a data source that is configured under the data source name User's Data using the alias USER1. The first example uses the connection method that is guaranteed to be present at the lowest level of ODBC conformance. DATASRC= names can contain quotation marks and spaces.

```
proc sql;
  connect to ODBC as user1
  (datasrc="User's Data" user=myusr1 password=mypwd1);
```

This example uses the connection method that represents a more advanced level of ODBC conformance. It uses the input dialog box that is provided by the driver. The DATASRC= and USER= arguments are within the connection string. The SQL pass-through facility therefore does not parse them but instead passes them to the ODBC manager.

```
proc sql;
  connect to odbc as user1
  (required="dsn=User's Data;uid=myusr1");
```

This example enables you to select any data source that is configured on your machine. The example uses the connection method that represents a more advanced level of ODBC conformance, Level 1. When connection succeeds, the connection string is returned in the SQLXMSG and SYSDBMSG macro variables. The connection string can be stored if this method is used to configure a connection for later use.

```
proc sql;
  connect to odbc (required);
```

This next example prompts you to specify the information that is required to make a connection to the DBMS. You are prompted to provide the data source name, user ID, and password in the dialog boxes that are displayed.

```
proc sql;
  connect to odbc (prompt);
```

## Connection to Component Examples

This example sends an Oracle SQL query (presented in highlighted text) to the Oracle database for processing. The results from the query serve as a virtual table for the PROC SQL FROM clause. In this example, MYCON is a connection alias.

```
proc sql;
connect to odbc as mycon
  (datasrc=mysrv1 user=myusr1 password=mypwd1);

select *
  from connection to mycon
    (select empid, lastname, firstname,
      hiredate, salary
      from sasdemo.employees
```

```

where hiredate>='31.12.1988');

disconnect from mycon;
quit;

```

This next example gives the previous query a name and stores it as the SQL view Samples.Hires88. The CREATE VIEW statement appears highlighted.

```

libname samples 'SAS-library';

proc sql;
connect to odbc as mycon
  (datasrc=mysrv1 user=myusr1 password=mypwd1);

create view samples.hires88 as
  select *
    from connection to mycon
      (select empid, lastname, firstname,
             hiredate, salary from sasdemo.employees
              where hiredate>='31.12.1988');

disconnect from mycon;
quit;

```

This example connects to Microsoft Access and creates a view NEWORDERS from all columns in the ORDERS table.

```

proc sql;
  connect to odbc as mydb
    (datasrc=MSAccess7);
  create view neworders as
    select * from connection to mydb
      (select * from orders);
  disconnect from mydb;
quit;

```

This next example sends an SQL query to Microsoft SQL Server, configured under the data source name **SQL Server**, for processing. The results from the query serve as a virtual table for the PROC SQL FROM clause.

```

proc sql;
  connect to odbc as mydb
    (datasrc="SQL Server" user=myusr1 password=mypwd1);
  select * from connection to mydb
    (select CUSTOMER, NAME, COUNTRY
      from CUSTOMERS
      where COUNTRY <> 'USA');
quit;

```

This example returns a list of the columns in the CUSTOMERS table.

```

proc sql;
  connect to odbc as mydb
    (datasrc="SQL Server" user=myusr1 password=mypwd1);
  select * from connection to mydb
    (ODBC::SQLColumns (, , "CUSTOMERS"));
quit;

```

## Special Catalog Queries

SAS/ACCESS Interface to ODBC supports the following special queries. Many databases provide or use system tables that allow queries to return the list of available tables, columns, procedures, and other useful information. ODBC provides much of this functionality through special application programming interfaces (APIs) to accommodate databases that do not follow the SQL table structure. You can use these special queries on SQL and non-SQL databases.

Here is the general format of the special queries:

`ODBC::SQLAPI "parameter 1","parameter n"`

`ODBC::`

required to distinguish special queries from regular queries.

`SQLAPI`

is the specific API that is being called. Neither `ODBC::` nor `SQLAPI` are case sensitive.

`"parameter n"`

a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters; the underscore represents any single character. Each driver also has an escape character that can be used to place characters within the string. See the driver documentation to determine the valid escape character.

The values for the special query arguments are DBMS-specific. For example, you provide the fully qualified table name for a `"Catalog"` argument. In dBase, the value of `"Catalog"` might be `c:\dbase\tst.dbf` and in Microsoft SQL Server, the value might be `test.customer`. In addition, depending on the DBMS that you are using, valid values for a `"Schema"` argument might be a user ID, a database name, or a library. All arguments are optional. If you specify some but not all arguments within a parameter, use a comma to indicate the omitted arguments. If you do not specify any parameters, commas are not necessary. Special queries are not available for all ODBC drivers.

ODBC supports these special queries:

`ODBC::SQLColumns <"Catalog", "Schema", "Table-name", "Column-name">`

returns a list of all columns that match the specified arguments. If no arguments are specified, all accessible column names and information are returned.

`ODBC::SQLColumnPrivileges <"Catalog", "Schema", "Table-name", "Column-name">`

returns a list of all column privileges that match the specified arguments. If no arguments are specified, all accessible column names and privilege information are returned.

`ODBC::SQLDataSources`

returns a list of database aliases to which ODBC is connected.

`ODBC::SQLDBMSInfo`

returns a list of DB2 databases (DSNs) to which ODBC is connected. It returns one row with two columns that describe the DBMS name (such as Microsoft SQL Server or Oracle) and the corresponding DBMS version.

- ODBC::SQLForeignKeys <"PK-catalog", "PK-schema", "PK-table-name", "FK-catalog", "FK-schema", "FK-table-name">**  
 returns a list of all columns that comprise foreign keys that match the specified arguments. If no arguments are specified, all accessible foreign key columns and information are returned.
- ODBC::SQLGetTypeInfo**  
 returns information about the data types that are supported in the data source.
- ODBC::SQLPrimaryKeys <"Catalog", "Schema", "Table-name">**  
 returns a list of all columns that compose the primary key that matches the specified table. A primary key can be composed of one or more columns. If no table name is specified, this special query fails.
- ODBC::SQLProcedures <"Catalog", "Schema", "Procedure-name">**  
 returns a list of all procedures that match the specified arguments. If no arguments are specified, all accessible procedures are returned.
- ODBC::SQLProcedureColumns <"Catalog", "Schema", "Procedure-name", "Column-name">**  
 returns a list of all procedure columns that match the specified arguments. If no arguments are specified, all accessible procedure columns are returned.
- ODBC::SQLSpecialColumns <"Identifier-type", "Catalog-name", "Schema-name", "Table-name", "Scope", "Nullable">**  
 returns a list of the optimal set of columns that uniquely identify a row in the specified table.
- ODBC::SQLStatistics <"Catalog", "Schema", "Table-name">**  
 returns a list of the statistics for the specified table name. You can specify the SQL\_INDEX\_ALL and SQL\_ENSURE options in the SQLStatistics API call. If the table name argument is not specified, this special query fails.
- ODBC::SQLTables <"Catalog", "Schema", "Table-name", "Type">**  
 returns a list of all tables that match the specified arguments. If no arguments are specified, all accessible table names and information are returned.
- ODBC::SQLTablePrivileges <"Catalog", "Schema", "Table-name">**  
 returns a list of all tables and associated privileges that match the specified arguments. If no arguments are specified, all accessible table names and associated privileges are returned.

# Autopartitioning Scheme for ODBC

## Overview

Autopartitioning for SAS/ACCESS Interface to ODBC is a modulo (MOD) function method. For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

## Autopartitioning Restrictions

SAS/ACCESS Interface to ODBC places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- SQL\_INTEGER, SQL\_BIT, SQL\_SMALLINT, and SQL\_TINYINT columns are given preference.
- You can use SQL\_DECIMAL, SQL\_DOUBLE, SQL\_FLOAT, SQL\_NUMERIC, and SQL\_REAL columns for partitioning under these conditions:
  - The ODBC driver supports converting these types to SQL\_INTEGER by using the INTEGER cast function.
  - The precision minus the scale of the column is greater than 0 but less than 10—that is,  $0 < \text{precision-scale} < 10$ .

The exception to the above rule is for Oracle SQL\_DECIMAL columns. As long as the scale of the SQL\_DECIMAL column is 0, you can use the column as the partitioning column.

## Nullable Columns

If you select a nullable column for autopartitioning, the OR<column-name>IS NULL SQL statement is appended at the end of the SQL code that is generated for the threaded Read. This ensures that any possible NULL values are returned in the result set. Also, if the column to be used for the partitioning is SQL\_BIT, the number of threads are automatically changed to two, regardless of how the DBSLICEPARM= option is specified.

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in the WHERE clause. For example, the following DATA step could not use a threaded Read to retrieve the data because all numeric columns in the table are in the WHERE clause.

```
data work.locomp;
  set trlib.MYEMPS;
  where EMPNUM<=30 and ISTENURE=0 and
        SALARY<=35000 and NUMCLASS>2;
run;
```

---

## Using DBSLICEPARM=

SAS/ACCESS Interface to ODBC defaults to three threads when you use autopartitioning. However, do not specify a maximum number of threads in **DBSLICEPARM=** to use for the threaded Read.

---

## Using DBSLICE=

You might achieve the best possible performance when using threaded Reads by specifying the **DBSLICE=** option for ODBC in your SAS operation. This is especially true if your DBMS supports multiple database partitions and provides a mechanism to allow connections to individual partitions. If your DBMS supports this concept, you can configure an ODBC data source for each partition. You can also use the **DBSLICE=** clause to specify both the data source and the WHERE clause for each partition, as shown in this example.

```
proc print data=trllib.MYEMPS (DBSLICE=(DSN1="EMPNUM BETWEEN 1 AND 33"
   DSN2="EMPNUM BETWEEN 34 AND 66"
   DSN3="EMPNUM BETWEEN 67 AND 100"));
run;
```

See your DBMS or ODBC driver documentation for more information about configuring for multiple partition access. You can also see “[Configuring Microsoft SQL Server Partitioned Views for Use with DBSLICE=](#)” on page 1089 for an example of configuring multiple partition access to a table.

Using the **DATASOURCE=** syntax is not required to use **DBSLICE=** with threaded Reads for the ODBC interface. The methods and examples described in **DBSLICE=** work well in cases where the table that you want to read is not stored in multiple partitions in your DBMS. These methods also give you flexibility in column selection. For example, if you know that the STATE column in your employee table contains only a few distinct values, you can customize your **DBSLICE=** clause accordingly.

```
datawork.locemp;
set trllib2.MYEMP (DBSLICE= ("STATE='FL'" "STATE='GA'"
                             "STATE='SC'" "STATE='VA'" "STATE='NC'"));
where EMPNUM<=30 and ISTENURE=0 and SALARY<=35000 and NUMCLASS>2;
run;
```

---

## Configuring Microsoft SQL Server Partitioned Views for Use with DBSLICE=

Microsoft SQL Server implements multiple partitioning by creating a global view across multiple instances of a Microsoft SQL Server database. For this example, assume that Microsoft SQL Server has been installed on three separate machines (SERVER1, SERVER2, SERVER3). Three ODBC data sources (SSPART1, SSPART2, SSPART3) have been configured against these servers. Also, a linked

server definition for each of these servers has been specified. This example uses SAS to create the tables and associated views, but you can create the tables and associated outside of the SAS environment.

**1 Create a local SAS table to build the Microsoft SQL Server tables.**

```
data work.MYEMPS;
format HIREDATE mmddyy 0. SALARY 9.2
      NUMCLASS 6. GENDER $1. STATE $2. EMPNUM 10. ;
do EMPNUM=1 to 100;
  morf=mod(EMPNUM,2)+1;
  if(morf eq 1) then
    GENDER='F';
  else
    GENDER='M';
  SALARY=(ranuni(0)*5000);
  HIREDATE=int(ranuni(13131)*3650);
  whatstate=int(EMPNUM/5);
  if(whatstate eq 1) then
    STATE='FL';
  if(whatstate eq 2) then
    STATE='GA';
  if(whatstate eq 3) then
    STATE='SC';
  if(whatstate eq 4) then
    STATE='VA';
  else
    state='NC';
  ISENURE=mod(EMPNUM,2);
  NUMCLASS=int(EMPNUM/5)+2;
  output;
end;
run;
```

**2 Create a table on each of the Microsoft SQL Server databases with the same table structure, and insert one-third of the overall data into each table.**

```
libname trlib odbc user=ssuser pw=sspwd dsn=sspart1;
proc datasets library=trlib;
  delete MYEMPS1;run;
run;
data trlib.MYEMPS1(drop=morf whatstate
DBTYPE=(HIREDATE="datetime" SALARY="numeric(8,2)"
NUMCLASS="smallint" GENDER="char(1)" ISENURE="bit" STATE="char(2)"
EMPNUM="int NOT NULL Primary Key CHECK (EMPNUM BETWEEN 0 AND 33)");
set work.MYEMPS;
where (EMPNUM BETWEEN 0 AND 33);
run;

libname trlib odbc user=ssuer pw=sspwd dsn=sspart2;
proc datasets library=trlib;
  delete MYEMPS2;run;
data trlib.MYEMPS2(drop=morf whatstate
DBTYPE=(HIREDATE="datetime" SALARY="numeric(8,2)"
NUMCLASS="smallint" GENDER="char(1)" ISENURE="bit" STATE="char(2)"
EMPNUM="int NOT NULL Primary Key CHECK (EMPNUM BETWEEN 34 AND 66)");
set work.MYEMPS;
where (EMPNUM BETWEEN 34 AND 66);
```

```

run;

libname trlib odbc user=ssuer pw=sspwd dsn=sspart3;
proc datasets library=trlib;
  delete MYEMPS3;run;
data trlib.MYEMPS3 (drop=morf whatstate
  DBTYPE=(HIREDATE="datetime" SALARY="numeric(8,2)"
  NUMCLASS="smallint" GENDER="char(1)" ISENTERURE="bit" STATE="char(2)"
  EMPNUM="int NOT NULL Primary Key CHECK (EMPNUM BETWEEN 67 AND 100)"));
  set work.MYEMPS;
  where (EMPNUM BETWEEN 67 AND 100);
run;

```

These table definitions also use CHECK constraints to enforce the distribution of the data on each of the subtables of the target view.

- 3 Create a view using the UNION ALL construct on each Microsoft SQL Server instance that references the other two tables.

```

/*SERVER1,SSPART1*/
proc sql noerrorstop;
connect to odbc (UID=ssuser PWD=sspwd DSN=SSPART1);
execute (drop view MYEMPS) by odbc;
execute (create view MYEMPS AS
  SELECT * FROM users.ssuser.MYEMPS1
  UNION ALL
  SELECT * FROM SERVER2.users.ssuser.MYEMPS2
  UNION ALL
  SELECT * FROM SERVER3.users.ssuser.MYEMPS3) by odbc;
quit;

/*SERVER2,SSPART2*/
proc sql noerrorstop;
connect to odbc (UID=ssuser PWD=sspwd DSN=SSPART2);
execute (drop view MYEMPS) by odbc;
execute (create view MYEMPS AS
  SELECT * FROM users.ssuser.MYEMPS2
  UNION ALL
  SELECT * FROM SERVER1.users.ssuser.MYEMPS1
  UNION ALL
  SELECT * FROM SERVER3.users.ssuser.MYEMPS3) by odbc;
quit;

/*SERVER3,SSPART3*/
proc sql noerrorstop;
connect to odbc (UID=ssuser PWD=sspwd DSN=SSPART3);
execute (drop view MYEMPS) by odbc;
execute (create view MYEMPS AS
  SELECT * FROM users.ssuser.MYEMPS3
  UNION ALL
  SELECT * FROM SERVER2.users.ssuser.MYEMPS2
  UNION ALL
  SELECT * FROM SERVER1.users.ssuser.MYEMPS1) by odbc;
quit;

```

This creates a global view that references the entire data set.

- 4 Set up your SAS operation to perform the threaded Read.

```

proc print data=trlib.MYEMPS(DBLICE=(sspart1="EMPNUM BETWEEN 1 AND 33"
sspart2="EMPNUM BETWEEN 34 AND 66"
sspart3="EMPNUM BETWEEN 67 AND 100"));
run;

```

The DBSLICE= option contains the Microsoft SQL Server partitioning information.

This configuration lets the ODBC interface access the data for the MYEMPS view directly from each subtable on the corresponding Microsoft SQL Server instance. The data is inserted directly into each subtable, but this process can also be accomplished by using the global view to divide up the data. For example, you can create empty tables and then create the view as seen in the example with the UNION ALL construct. You can then insert the data into the view MYEMPS. The CHECK constraints allow the Microsoft SQL Server query processor to determine which subtables should receive the data.

Other tuning options are available when you configure Microsoft SQL Server to use partitioned data. For more information, see the "Creating a Partitioned View" and "Using Partitioned Views" sections in *Creating and Maintaining Databases (SQL Server 2000)*.

## DBLOAD Procedure Specifics for ODBC

### Overview

For general information about this feature, see the [Appendix 3, “DBLOAD Procedure”](#). ODBC examples are available.

SAS/ACCESS Interface to ODBC supports all [DBLOAD procedure statements](#) (except ACCDESC=) in batch mode. Here are the DBLOAD procedure specifics for ODBC:

- The DBLOAD step DBMS= value is ODBC.
- Here are the database description statements that PROC DBLOAD uses:

**DSN= <'>ODBC-data-source<'>;**  
specifies the name of the data source in which you want to store the new ODBC table. The *data-source* is limited to eight characters.

The data source that you specify must already exist. If the data source name contains the \_, \$, @, or # special character, you must enclose it in quotation marks. The ODBC standard recommends against using special characters in data source names, however.

**USER= <'>ODBC-user name<'>;**  
lets you connect to an ODBC database with a user ID that is different from the default ID. USER= is optional in ODBC. If you specify USER=, you must also specify PASSWORD=. If USER= is omitted, your default user ID is used.

**PASSWORD=<'>ODBC-password<'>;**  
specifies the ODBC password that is associated with your user ID.

PASSWORD= is optional in ODBC because users have default user IDs. If you specify USER=, you must specify PASSWORD=.

---

**Note:** If you do not want to enter your ODBC password in uncoded text on this statement, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

---

BULKCOPY= YES|NO;

determines whether SAS uses the Microsoft Bulk Copy facility to insert data into a DBMS table (Microsoft SQL Server only). The default value is NO.

The Microsoft Bulk Copy (BCP) facility lets you efficiently insert rows of data into a DBMS table as a unit. As the ODBC interface sends each row of data to BCP, the data is written to an input buffer. When you have inserted all rows or the buffer reaches a certain size, all rows are inserted as a unit into the table, and the data is committed to the table. The DBCOMMIT= data set option determines the size of the buffer.

You can also set the DBCOMMIT=*n* option to commit rows after every *n* insertions.

If an error occurs, a message is written to the SAS log, and any rows that have been inserted in the table before the error are rolled back.

---

**Note:** BULKCOPY= is not supported on UNIX.

---

- Here is the TABLE= statement:

TABLE= <*authorization-id*.>*table-name*;

identifies the table or view that you want to use to create an access descriptor. The TABLE= statement is required.

The *authorization-id* is a user ID or group ID that is associated with the table.

- Here is the NULLS statement:

NULLS *variable-identifier-1* =Y|N|D < . . . *variable-identifier-n* =Y|N|D >;

enables you to specify whether the columns that are associated with the listed SAS variables allow NULL values. By default, all columns accept NULL values.

The NULLS statement accepts any one of these three values:

- Y – specifies that the column accepts NULL values. This is the default.
- N – specifies that the column does not accept NULL values.
- D – specifies that the column is specified as NOT NULL WITH DEFAULT.

## Examples

This example creates a new ODBC table, MYUSR1.EXCHANGE, from the DLIB.RATEOFEX data file. You must be granted the appropriate privileges in order to create new ODBC tables or views.

```
proc dbload dbms=odbc data=dlib.rateofex;
  dsn=sample;
```

```

user='myusr1';
password='mypwd1';
table=exchange;
rename fgnindol=fgnindollars
        4=dollarsinfgn;
nulls updated=n fgnindollars=n
        dollarsinfgn=n country=n;
load;
run;

```

This next example sends only an ODBC SQL GRANT statement to the SAMPLE database and does not create a new table. Therefore, the TABLE= and LOAD statements are omitted.

```

proc dbload dbms=odbc;
    user='myusr1';
    password='mypwd1';
    dsn=sample;
    sql grant select on myusr1.exchange
        to testcase;
run;

```

## Temporary Table Support for ODBC

SAS/ACCESS Interface to ODBC supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## Passing SAS Functions to ODBC

SAS/ACCESS Interface to ODBC passes the following SAS functions to the data source for processing if the DBMS server supports this function. Where the ODBC function name differs from the SAS SQL function name, the ODBC name appears in parentheses. For details, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|       |         |
|-------|---------|
| ABS   | LOG10   |
| ARCOS | LOWCASE |
| ARSIN | MAX     |
| ATAN  | MIN     |
| AVG   | SIGN    |
| CEIL  | SIN     |
| COS   | SQRT    |
| COT   | STRIP   |
| COUNT | SUM     |
| EXP   | TAN     |

FLOOR    UPCASE  
LOG

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to ODBC. Due to incompatibility in date and time functions between ODBC and SAS, ODBC might not process them correctly. Check your results to determine whether these functions are working as expected.

|                    |                     |
|--------------------|---------------------|
| BYTE (CHAR)        | REPEAT              |
| COMPRESS (REPLACE) | SECOND              |
| DATE (CURDATE)     | SOUNDEX             |
| DATEPART           | SUBSTR (SUBSTRING)  |
| DATETIME (NOW)     | TIME (CURTIME)      |
| DAY (DAYOFMONTH)   | TIMEPART            |
| HOUR               | TODAY (CURDATE)     |
| INDEX (LOCATE)     | TRIMN (RTRIM)       |
| LENGTH             | TRANWRD (REPLACE)   |
| MINUTE             | WEEKDAY (DAYOFWEEK) |
| MONTH              | YEAR                |
| QTR (QUARTER)      |                     |

## Passing Joins to ODBC

For a multiple libref join to pass to ODBC, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- data source (DATASRC=)
- catalog (QUALIFIER=)
- update isolation level (UPDATE\_ISOLATION\_LEVEL=, if specified)
- read isolation level (READ\_ISOLATION\_LEVEL=, if specified)
- prompt (PROMPT=, must *not* be specified)

Outer joins between two or more tables can be passed to the DBMS. However, the outer joins must not be mixed with inner joins in a query.

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Bulk Loading for ODBC

The [BULKLOAD=](#) on page 171 LIBNAME option calls the Bulk Copy (BCP) facility, which lets you efficiently insert rows of data into a DBMS table as a unit. BCP= is an alias for this option.

**Windows Specifics:** The Bulk Copy facility is available only when you are accessing Microsoft SQL Server data on Windows platforms. To use this facility, your installation of Microsoft SQL Server must include the ODBCBCP.DLL file. BULKCOPY= is not available on UNIX.

As the ODBC interface sends rows of data to the Bulk Copy facility, data is written to an input buffer. When you send all rows or when the buffer reaches a certain size ([DBCOMMIT=](#) determines this), all rows are inserted as a unit into the table and the data is committed to the table. You can also set DBCOMMIT= to commit rows after a specified number of rows are inserted.

If an error occurs, a message is written to the SAS log, and any rows that were inserted before the error are rolled back.

---

## Locking in the ODBC Interface

The following LIBNAME and data set options let you control how the ODBC interface handles locking when a table is read. For general information about an option, see ["LIBNAME Options for Relational Databases"](#) on page 134.

**READ\_LOCK\_TYPE=**ROW | TABLE | NOLOCK

ROW locks a row if any of its columns are accessed during a read transaction.

TABLE locks an entire table.

NOLOCK does not lock a DBMS table or any rows during a read transaction.

**UPDATE\_LOCK\_TYPE=** ROW | TABLE | NOLOCK

ROW locks a row if any of its columns are going to be updated.

TABLE locks an entire table.

NOLOCK does not lock a DBMS table or any rows when reading them for an update.

**READ\_ISOLATION\_LEVEL=** S | RR | RC | RU | V

The ODBC driver manager supports the S, RR, RC, RU, and V isolation levels that are defined in this table.

**Table 30.3** Isolation Levels for ODBC

| Isolation Level       | Definition                                                                                                                                                                                                                      |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| S (serializable)      | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.                                                                                                                                                              |
| RR (repeatable read)  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.                                                                                                                                                    |
| RC (read committed)   |                                                                                                                                                                                                                                 |
| RU (read uncommitted) | Allows dirty Reads, nonrepeatable Reads, and phantom Reads.                                                                                                                                                                     |
| V (versioning)        | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads. These transactions are serializable but higher concurrency is possible than with the serializable isolation level. Typically, a nonlocking protocol is used. |

Here are how the terms in the table are defined.

#### *Dirty reads*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it can see changes that are made by those concurrent transactions even before they commit.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

#### *Nonrepeatable reads*

If a transaction exhibits this phenomenon, it might read a row once and later fail when it attempts to read that row again in the same transaction. The row might have been changed or even deleted by a concurrent transaction. Therefore, the read is not necessarily repeatable.

For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

#### *Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist, a phantom.

`UPDATE_ISOLATION_LEVEL= S | RR | RC | V`

The ODBC driver manager supports the S, RR, RC, and V isolation levels defined in the preceding table.

---

# Naming Conventions for ODBC

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Because ODBC is an application programming interface (API) rather than a database, table names and column names are determined at run time. Most SAS names can be up to 32 characters long. SAS/ACCESS Interface to ODBC supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, SAS truncates them to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).)

This example specifies SAP ASE as the DBMS.

```
libname mydblib odbc user=myusr1 password=mypwd1  
      database=sapase;  
  
data mydblib.a;  
      x=1;  
      y=2;  
run;
```

SAP ASE is generally case sensitive. This example would therefore produce an SAP ASE table named `a` with columns named `x` and `y`.

If the DBMS being accessed is not case sensitive, such as Oracle, the example would produce an Oracle table named `A` and columns named `x` and `y`. The object names would be normalized to uppercase.

---

# Data Types for ODBC

---

## Overview

Every column in a table has a name and a data type. The data type tells the DBMS how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about ODBC null and default values and data conversions.

## ODBC Null Values

Many relational database management systems have a special value called NULL. A DBMS NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a DBMS NULL value, it interprets it as a SAS missing value.

In most relational databases, columns can be specified as NOT NULL so that they require data (they cannot contain NULL values). When a column is specified as NOT NULL, the DBMS does not add a row to the table unless the row has a value for that column. When creating a DBMS table with SAS/ACCESS, you can use the **DBNULL=** data set option to indicate whether NULL is a valid value for specified columns.

ODBC mirrors the behavior of the underlying DBMS with regard to NULL values. See the documentation for your DBMS for information about how it handles NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how the DBMS handles SAS missing character values, use the **NULLCHAR=** and **NULLCHARVAL=** data set options.

## LIBNAME Statement Data Conversions

This table shows all data types and default SAS formats that SAS/ACCESS Interface to ODBC supports. It does not explicitly specify the data types as they exist for each DBMS. It lists the SQL types that each DBMS data type would map to. For example, a CHAR data type under DB2 would map to an ODBC data type of SQL\_CHAR. All data types are supported.

*Table 30.4 ODBC Data Types and Default SAS Formats*

| ODBC Data Type    | Default SAS Format |
|-------------------|--------------------|
| SQL_CHAR          | \$w.               |
| SQL_VARCHAR       | \$w.               |
| SQL_LONGVARCHAR   | \$w.               |
| SQL_BINARY        | \$w. <sup>1</sup>  |
| SQL_VARBINARY     | \$w. <sup>1</sup>  |
| SQL_LONGVARBINARY | \$w. <sup>1</sup>  |

| ODBC Data Type     | Default SAS Format                                                 |
|--------------------|--------------------------------------------------------------------|
| SQL_DECIMAL        | w. or w.d or none if w and d are not specified                     |
| SQL_NUMERIC        | w. or w.d or none if w and d are not specified                     |
| SQL_INTEGER        | 11.                                                                |
| SQL_SMALLINT       | 6.                                                                 |
| SQL_TINYINT        | 4.                                                                 |
| SQL_BIT            | 1.                                                                 |
| SQL_REAL           | none                                                               |
| SQL_FLOAT          | none                                                               |
| SQL_DOUBLE         | none                                                               |
| SQL_BIGINT         | 20.                                                                |
| SQL_INTERVAL       | \$w.                                                               |
| SQL_GUID           | \$w.                                                               |
| SQL_TYPE_DATE      | DATE9.                                                             |
| SQL_TYPE_TIME      | TIME8.<br>ODBC cannot support fractions of seconds for time values |
| SQL_TYPE_TIMESTAMP | DATETIMEw.d where w and d depend on precision                      |

**1** Because the ODBC driver does the conversion, this field is displayed as if the \$HEXw. format were applied.

This table shows the default data types that SAS/ACCESS Interface to ODBC uses when creating tables. SAS/ACCESS Interface to ODBC lets you specify non-default data types by using the **DBTYPE=** data set option.

Table 30.5 Default ODBC Output Data Types

| SAS Variable Format | Default ODBC Data Type                                    |
|---------------------|-----------------------------------------------------------|
| m.n                 | SQL_DOUBLE or SQL_NUMERIC using m.n if the DBMS allows it |

| SAS Variable Format | Default ODBC Data Type     |
|---------------------|----------------------------|
| \$n.                | SQL_VARCHAR using <i>n</i> |
| datetime formats    | SQL_TIMESTAMP              |
| date formats        | SQL_DATE                   |
| time formats        | SQL_TIME                   |

---

# Sample Programs for ODBC

Sample programs for SAS/ACCESS Interface to ODBC are available from <https://github.com/sassoftware/sas-access-samples>.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to OLE DB

---

|                                                               |      |
|---------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to OLE DB</i> | 1104 |
| <i>Introduction to SAS/ACCESS Interface to OLE DB</i>         | 1104 |
| <i>LIBNAME Statement for the OLE DB Engine</i>                | 1104 |
| Overview                                                      | 1104 |
| Arguments                                                     | 1105 |
| Connecting with OLE DB Services                               | 1110 |
| Connecting Directly to a Data Provider                        | 1110 |
| OLE DB LIBNAME Statement Examples                             | 1111 |
| <i>Data Set Options for OLE DB</i>                            | 1112 |
| <i>SQL Pass-Through Facility Specifics for OLE DB</i>         | 1114 |
| Key Information                                               | 1114 |
| Examples                                                      | 1115 |
| Special Catalog Queries                                       | 1115 |
| <i>Temporary Table Support for OLE DB</i>                     | 1119 |
| <i>Passing SAS Functions to OLE DB</i>                        | 1119 |
| <i>Passing Joins to OLE DB</i>                                | 1120 |
| <i>Bulk Loading for OLE DB</i>                                | 1121 |
| <i>Locking in the OLE DB Interface</i>                        | 1121 |
| <i>Accessing OLE DB for OLAP Data</i>                         | 1123 |
| Overview                                                      | 1123 |
| Using the SQL Pass-Through Facility with OLAP Data            | 1123 |
| <i>Naming Conventions for OLE DB</i>                          | 1125 |
| <i>Data Types for OLE DB</i>                                  | 1126 |
| Overview                                                      | 1126 |
| OLE DB Null Values                                            | 1126 |
| Specifying the BOOL_VAL Environment Variable                  | 1127 |
| LIBNAME Statement Data Conversions                            | 1127 |
| <i>Sample Programs for OLE DB</i>                             | 1129 |

---

# System Requirements for SAS/ACCESS Interface to OLE DB

You can find information about system requirements for SAS/ACCESS Interface to OLE DB in the following locations:

- [System Requirements for SAS/ACCESS Interface to OLE DB with SAS 9.4](#)
- [Third-Party Software Requirements for Use with SAS 9.4](#)

---

# Introduction to SAS/ACCESS Interface to OLE DB

For available SAS/ACCESS features, see [OLE DB supported features on page 111](#). For more information about OLE DB, see your OLE DB documentation.

Microsoft OLE DB is an application programming interface (API) that provides access to data that can be in a database table, an email file, a text file, or another type of file. This SAS/ACCESS interface accesses data from these sources through OLE DB data providers such as Microsoft Access, Microsoft SQL Server, and Oracle.

---

**Note:** SAS/ACCESS Interface to OLE DB is not included with SAS Viya.

---

---

# LIBNAME Statement for the OLE DB Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to OLE DB supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing OLE DB.

**LIBNAME** *libref oledb <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in *SAS Global Statements: Reference*.

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *oledb*

specifies the SAS/ACCESS engine name for the OLE DB interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the data source. You can connect to a data source either by using OLE DB Services or by connecting directly to the provider. For details, see “[Connecting with OLE DB Services](#)” on page 1110 and “[Connecting Directly to a Data Provider](#)” on page 1110.

These connection options are available with both connection methods. Here is how they are defined.

#### **USER=<'>OLE-DB-user-name<'>**

lets you connect to an OLE DB data source with a user ID that is different from the default ID. The default is your user ID.

#### **PASSWORD=<'>OLE-DB-password<'>**

specifies the OLE DB password that is associated with your user ID. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. If you do not wish to enter your OLE DB password in uncoded text, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

#### **DATASOURCE=<'>data-source<'>**

identifies the data source object (such as a relational database server or a local file) to which you want to connect.

#### **PROVIDER=<'>provider-name<'>**

specifies which OLE DB provider to use to connect to the data source. This option is required during batch processing. There is no restriction on the length of the *provider-name*. If the *provider-name* contains blank spaces or special characters, enclose it in quotation marks. If you do not specify a provider, an **OLE DB Services** dialog box prompts you for connection information. In batch mode, if you do not specify a provider the connection fails. To use the Ace, you must specify Microsoft.ACE.OLEDB.12.0. If you are using the Microsoft Jet OLE DB 4.0 provider, specify PROVIDER=JET.

#### **PROPERTIES=(<'>property-1<'>=<'>value-1<'> < . . . <'>property-n<'>=<'>value-n<'>)**

specifies standard provider properties that enable you to connect to a data source and to specify connection attributes. If a property name or value contains embedded spaces or special characters, enclose the name or value in quotation marks. Use a blank space to separate multiple properties. If your provider supports a password property, that value cannot be encoded. To use

an encoded password, use the `PASSWORD=` option instead. See your provider documentation for a list and description of all properties that your provider supports. No properties are specified by default.

**PROVIDER\_STRING=<'>extended-properties<'>**

specifies provider-specific extended connection information, such as the file type of the data source. If the string contains blank spaces or special characters, enclose it in quotation marks. For example, the Ace provider accepts strings that indicate file type, such as 'Excel 8.0'. The following example uses the Ace provider to access the spreadsheet `Y2KBUDGET.XLS`. Specify the 'Excel 8.0' provider string so that Ace recognizes the file as an Excel 8.0 worksheet.

```
libname budget oledb provider='Microsoft.ACE.OLEDB.12.0'
provider_string='Excel 8.0' datasource='d:\excel80\Y2Kbudget.xls';
```

**OLEDB\_SERVICES=YES | NO**

determines whether SAS uses OLE DB Services to connect to the data source. Specify YES to use OLE DB Services or specify NO to use the provider to connect to the data source. When you specify `PROMPT=YES` and `OLEDB_SERVICES=YES`, you can set more options than you would otherwise be able to specify when you are prompted by the provider's dialog box. If `OLEDB_SERVICES=NO`, you must specify `PROVIDER=` first so that the provider's prompt dialog boxes are used. If `PROVIDER=` is omitted, SAS uses OLE DB Services, even if you specify `OLEDB_SERVICES=NO`. YES is the default. When `OLEDB_SERVICES=YES` and a successful connection is made, the complete connection string is returned in the `SYSDBMSG` macro variable.

**PROMPT=YES | NO**

determines whether one of these interactive dialog boxes is displayed in SAS Display Manager to guide you through the connection process:

- an **OLE DB provider** dialog box if `OLEDB_SERVICES=NO` and you specify a provider.
- an **OLE DB Services** dialog box if `OLEDB_SERVICES=YES` or if you do not specify a provider.

Generally preferred over the provider's dialog box, the **OLE DB Services** dialog box lets you specify options more easily. If you specify a provider and set `OLEDB_SERVICES=NO`, the default is `PROMPT=NO`. Otherwise, the default is `PROMPT=YES`. If `OLEDB_SERVICES=YES` or if you do not specify a provider, an **OLE DB Services** dialog box is displayed even if you specify `PROMPT=NO`. Specify no more than one of the following options in each LIBNAME statement: `COMPLETE=`, `REQUIRED=`, `PROMPT=`. Any properties that you specify in the `PROPERTIES=` option are displayed in the prompting interface, and you can edit any field.

**UDL\_FILE=<'>path-and-file-name<'>**

specifies the path and file name for a Microsoft universal data link (UDL). For example, you could specify

`UDL_FILE="C:\WinNT\profiles\me\desktop\MyDBLink.udl"`. This option does not support SAS filerrefs. `SYSDBMSG` is not specified on successful completion. For more information, see Microsoft documentation about the Data Link API. This option overrides any values that are specified with the `INIT_STRING=`, `PROVIDER=`, and `PROPERTIES=` options.

This connection option is available only when you use OLE DB Services.

**INIT\_STRING='property-1=value-1<...;property-n=value-n'**  
 specifies an initialization string, enabling you to bypass the interactive prompting interface yet still use OLE DB Services. (This option is not available if OLEDB\_SERVICES=NO.) Use a semicolon to separate properties. After you connect to a data source, SAS returns the complete initialization string to the macro variable SYSDBMSG, which stores the connection information that you specify in the prompting window. You can reuse the initialization string to make automated connections or to specify connection information for batch jobs. For example, assume that you specify this initialization string:

```
init_string='Provider=SQLOLEDB;Password=dbmgr1;Persist  
Security Info=True;User ID=rachel;Initial Catalog=users;  
Data Source=dwtsrv1';
```

Here is what the content of the SYSDBMSG macro variable would be:

```
OLEDB: Provider=SQLOLEDB;Password=dbmgr1;  
Persist Security Info=True;User ID=rachel;  
Initial Catalog=users;Data Source=dwtsrv1;
```

If you store this string for later use, delete the OLEDB: prefix and any initial spaces before the first listed option. There is no default value. However, if you specify a null value for this option, the OLE DB Provider for ODBC (MSDASQL) is used with your default data source and its properties. See your OLE DB documentation for more information about these default values. This option overrides any values that are specified with the PROVIDER= and PROPERTIES= options. To write the initialization string to the SAS log, submit this code immediately after connecting to the data source:

```
%put %superq(SYSDBMSG);
```

Only these connection options are available when you connect directly to a provider.

#### COMPLETE=YES | NO

specifies whether SAS attempts to connect to the data source without prompting you for connection information. If you specify COMPLETE=YES and the connection information that you specify in your LIBNAME statement is sufficient, then SAS makes the connection and does not prompt you for additional information. If you specify COMPLETE=YES and the connection information that you specify in your LIBNAME statement is not sufficient, the provider's dialog box prompts you for additional information. You can enter optional information as well as required information in the dialog box. NO is the default value. COMPLETE= is available only when you specify OLEDB\_SERVICES=NO and you specify a provider. It is not available in the SQL pass-through facility. Specify no more than one of these options in each LIBNAME statement: COMPLETE=, REQUIRED=, PROMPT=.

#### REQUIRED=YES | NO

specifies whether SAS attempts to connect to the data source without prompting you for connection information and whether you can interactively specify optional connection information. If you specify REQUIRED=YES and the connection information that you specify in your LIBNAME statement is sufficient, SAS makes the connection and you are not prompted for additional information. If you specify REQUIRED=YES and the connection information that you specify in your LIBNAME statement is not sufficient, the provider's dialog box prompts you for the required connection information. You cannot enter optional connection information in the dialog box. NO is the default value. REQUIRED= is available only when you specify OLEDB\_SERVICES=NO and you specify a provider. It is not available in the SQL pass-through facility. Specify no more than one of these options in each LIBNAME statement: COMPLETE=, REQUIRED=, PROMPT=.

option. It is not available in the SQL pass-through facility. Specify no more than one of these options in each LIBNAME statement: COMPLETE=, REQUIRED=, PROMPT=.

*LIBNAME-options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to OLE DB, with the applicable default values. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 31.1** SAS/ACCESS LIBNAME Options for OLE DB

| Option               | Default Value                                      |
|----------------------|----------------------------------------------------|
| ACCESS=              | none                                               |
| AUTHDOMAIN=          | none                                               |
| AUTOCOMMIT=          | data-source specific                               |
| BL_KEEPIDENTITY=     | NO                                                 |
| BL_KEEPNULLS=        | YES                                                |
| BL_OPTIONS=          | not specified                                      |
| BULKLOAD=            | NO                                                 |
| CELLPROP=            | VALUE                                              |
| CHAR_AS_NCHAR=       | NO                                                 |
| COMMAND_TIMEOUT=     | 0 (no time-out)                                    |
| CONNECTION=          | SHAREDREAD                                         |
| CONNECTION_GROUP=    | none                                               |
| CURSOR_TYPE=         | FORWARD_ONLY                                       |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows) |
| DBCONINIT=           | none                                               |
| DBCONTERM=           | none                                               |
| DBCREATE_TABLE_OPTS= | none                                               |
| DBGEN_NAME=          | DBMS                                               |

| <b>Option</b>             | <b>Default Value</b>                                  |
|---------------------------|-------------------------------------------------------|
| DBINDEX=                  | NO                                                    |
| DBLIBINIT=                | none                                                  |
| DBLIBTERM=                | none                                                  |
| DBMAX_TEXT=               | 1024                                                  |
| DBMSTEMP=                 | NO                                                    |
| DBNULLKEYS=               | YES                                                   |
| DEFER=                    | NO                                                    |
| DELETE_MULT_ROWS=         | NO                                                    |
| DIRECT_SQL=               | YES                                                   |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                    |
| INSERT_SQL=               | data-source specific                                  |
| INSERTBUFF=               | 1                                                     |
| MULTI_DATASRC_OPT=        | NONE                                                  |
| PRESERVE_COL_NAMES=       | see “Naming Conventions for OLE DB”                   |
| PRESERVE_GUID=            | none                                                  |
| PRESERVE_TAB_NAMES=       | see “Naming Conventions for OLE DB”                   |
| QUALIFIER=                | none                                                  |
| QUALIFY_ROWS=             | NO                                                    |
| QUOTE_CHAR=               | not specified                                         |
| READBUFF=                 | 1                                                     |
| READ_ISOLATION_LEVEL=     | not specified (see “Locking in the OLE DB Interface”) |
| READ_LOCK_TYPE=           | see “Locking in the OLE DB Interface”                 |
| REREAD_EXPOSURE=          | NO                                                    |

| Option                  | Default Value                                                           |
|-------------------------|-------------------------------------------------------------------------|
| SCHEMA=                 | none                                                                    |
| SPOOL=                  | YES                                                                     |
| SQL_FUNCTIONS=          | none                                                                    |
| STRINGDATES=            | NO                                                                      |
| UPDATE_ISOLATION_LEVEL= | not specified (see “ <a href="#">Locking in the OLE DB Interface</a> ”) |
| UPDATE_LOCK_TYPE=       | ROW                                                                     |
| UPDATE_MULT_ROWS=       | NO                                                                      |
| UTILCONN_TRANSIENT=     | NO                                                                      |

---

## Connecting with OLE DB Services

By default, SAS/ACCESS Interface to OLE DB uses OLE DB services because this is often the fastest and easiest way to connect to a data provider.

OLE DB Services provides performance optimizations and scaling features, including resource pooling. It also provides interactive prompting for the provider name and connection information.

Assume that you submit a simple LIBNAME statement, such as this one:

```
libname mydblib oledb;
```

SAS directs OLE DB Services to display a dialog box that contains tabs where you select the OLE DB provider for your database.

After you make a successful connection using OLE DB Services, you can retrieve the connection information and reuse it in batch jobs and automated connections. For more information, see the connection options INIT\_STRING= and OLEDB\_SERVICES=.

---

## Connecting Directly to a Data Provider

To connect to a data source, SAS/ACCESS Interface to OLE DB requires a provider name and provider-specific connection information such as the user ID, password, schema, or server name. If you know all of this information, you can connect directly to a provider without using [OLE DB services](#).

If you are connecting to Microsoft SQL Server and you specify the SAS/ACCESS BULKLOAD=YES option, you must connect directly to the provider by specifying this information:

- the name of the provider (PROVIDER=)
- any required connection information

After you connect to your provider, you can use the special OLE DB **PROVIDER\_INFO** query to make subsequent unprompted connections easier. You can submit this special query as part of a PROC SQL query to display all available provider names and properties. For an example, see “[Examples of Special OLE DB Queries](#)” on page 1118.

If you know only the provider name and you are running an interactive SAS session, you can be prompted for the provider's properties. Specify PROMPT=YES to direct the provider to prompt you for properties and other connection information. Each provider displays its own prompting interface.

If you run SAS in a batch environment, specify only USER=, PASSWORD=, DATASOURCE=, PROVIDER=, and PROPERTIES=.

## OLE DB LIBNAME Statement Examples

In this example, the libref MYDBLIB uses the SAS/ACCESS OLE DB engine to connect to a Microsoft SQL Server database.

```
libname mydblib oledb user=myusr1 password=mypwd1
datasource=dept203 provider=sqloledb properties=('initial catalog'=mgronly);
proc print data=mydblib.customers;
    where state='CA';
run;
```

In the next example, the libref MYDBLIB uses the SAS/ACCESS engine for OLE DB to connect to an Oracle database. Because prompting is enabled, you can review and edit the user, password, and data source information in a dialog box.

```
libname mydblib oledb user=myusr1 password=mypwd1 datasource=mydb.world
provider=msdaora prompt=yes;

proc print data=mydblib.customers;
    where state='CA';
run;
```

For this next example, you submit a basic LIBNAME statement, so an **OLE DB Services** dialog box prompts you for the provider name and property values.

```
libname mydblib oledb;
```

The advantage of being prompted is that you do not need to know any special syntax to specify the values for the properties. Prompting also enables you to set more options than you might when you connect directly to the provider (and do not use OLE DB Services).

# Data Set Options for OLE DB

All SAS/ACCESS data set options in this table are supported for OLE DB. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 31.2** SAS/ACCESS Data Set Options for OLE DB

| Option               | Default Value        |
|----------------------|----------------------|
| BL_KEEPIDENTITY=     | LIBNAME option value |
| BL_KEEPNULLS=        | LIBNAME option value |
| BL_OPTIONS=          | LIBNAME option value |
| BULKLOAD=            | NO                   |
| COMMAND_TIMEOUT=     | LIBNAME option value |
| CURSOR_TYPE=         | LIBNAME option value |
| DBCOMMIT=            | LIBNAME option value |
| DBCONDITION=         | none                 |
| DBCREATE_TABLE_OPTS= | LIBNAME option value |
| DBFORCE=             | NO                   |
| DBGEN_NAME=          | DBMS                 |
| DBINDEX=             | LIBNAME option value |
| DBKEY=               | none                 |
| DBLABEL=             | NO                   |
| DBLARGETABLE=        | none                 |
| DBMAX_TEXT=          | 1024                 |
| DBNULL=              | _ALL_=YES            |
| DBNULLKEYS=          | LIBNAME option value |

| Option                    | Default Value               |
|---------------------------|-----------------------------|
| DBSASLABEL=               | COMPAT                      |
| DBSASTYPE=                | see "Data Types for OLE DB" |
| DBTYPE=                   | see "Data Types for OLE DB" |
| ERRLIMIT=                 | 1                           |
| IGNORE_READ_ONLY_COLUMNS= | NO                          |
| INSERT_SQL=               | LIBNAME option value        |
| INSERTBUFF=               | LIBNAME option value        |
| NULLCHAR=                 | SAS                         |
| NULLCHARVAL=              | a blank character           |
| PRESERVE_COL_NAMES=       | LIBNAME option value        |
| QUALIFIER=                | LIBNAME option value        |
| READBUFF=                 | LIBNAME option value        |
| READ_ISOLATION_LEVEL=     | LIBNAME option value        |
| SASDATEFMT=               | not specified               |
| SCHEMA=                   | LIBNAME option value        |
| UPDATE_ISOLATION_LEVEL=   | LIBNAME option value        |
| UPDATE_LOCK_TYPE=         | LIBNAME option value        |
| UTILCONN_TRANSIENT=       | YES                         |

---

# SQL Pass-Through Facility Specifics for OLE DB

---

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the OLE DB interface.

- The *dbms-name* is `OLEDB`.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to OLE DB. If you use multiple simultaneous connections, you must use an *alias* to identify the different connections. If you do not specify an alias, the default alias, `OLEDB`, is used. The functionality of multiple connections to the same OLE DB provider might be limited by a particular provider.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#). For some data sources, the connection options have default values and are therefore not required.  
Not all OLE DB providers support all connection options. See your provider documentation for more information.
- Here are the LIBNAME options that are available with the CONNECT statement.
  - [AUTOCOMMIT=](#)
  - [CELLPROP=](#)
  - [COMMAND\\_TIMEOUT=](#)
  - [CURSOR\\_TYPE=](#)
  - [DBMAX\\_TEXT=](#)
  - [QUALIFY\\_ROWS=](#)
  - [READ\\_ISOLATION\\_LEVEL=](#)
  - [READ\\_LOCK\\_TYPE=](#)
  - [READBUFF=](#)
  - [STRINGDATES=](#)

---

## Examples

This example uses an alias to connect to a Microsoft SQL Server database and select a subset of data from the PAYROLL table. The SAS/ACCESS engine uses OLE DB Services to connect to OLE DB because this is the default action when the OLEDB\_SERVICES= option is omitted.

```
proc sql;
connect to oledb as finance
  (user=myusr1 password=mypwd1 datasource=dwtsrv1
   provider=sqloledb);

select * from connection to finance (select * from payroll
                                       where jobcode='FA3');
quit;
```

In this example, the CONNECT statement omits the provider name and properties. An **OLE DB Services** dialog box prompts you for the connection information.

```
proc sql;
connect to oledb;
quit;
```

This example uses OLE DB Services to connect to a provider that is configured under the data source name User's Data with the alias USER1. Note that the data source name can contain quotation marks and spaces.

```
proc sql;
  connect to oledb as user1
    (provider='Microsoft.ACE.OLEDB.12.0' datasource='c:\db1.mdb');;
```

---

## Special Catalog Queries

---

### Overview

SAS/ACCESS Interface to OLE DB supports the following special queries. Many databases provide or use system tables that allow queries to return the list of available tables, columns, procedures, and other useful information. OLE DB provides much of this functionality through special application programming interfaces (APIs) to accommodate databases that do not follow the SQL table structure. You can use these special queries on SQL and non-SQL databases.

Not all OLE DB providers support all queries. See your provider documentation for more information.

Here is the general format of the special queries:

OLEDDB::*schema-rowset*("parameter 1","parameter n")

OLEDDB::  
is required to distinguish special queries from regular queries.

***schema-rowset***

is the specific schema rowset that is being called. All valid schema rowsets are listed under the IDBSchemaRowset Interface in the *Microsoft OLE DB Programmer's Reference*. Both OLEDB:: and *schema-rowset* are case sensitive.

***"parameter n"***

is a quoted string that is enclosed by commas. The values for the special query arguments are specific to each data source. For example, you provide the fully qualified table name for a "Qualifier" argument. In dBase, the value of "Qualifier" might be c:\dbase\tst.dbf, and in SQL Server, the value might be test.customer. In addition, depending on the data source that you use, values for an "Owner" argument might be a user ID, a database name, or a library. All arguments are optional. If you specify some but not all arguments within a parameter, use commas to indicate omitted arguments. If you do not specify any parameters, no commas are necessary. These special queries might not be available for all OLE DB providers.

OLE DB supports these special queries:

OLEDB::ASSERTIONS( <"Catalog", "Schema", "Constraint-Name"> )

returns assertions that are specified in the catalog that a given user owns.

OLEDB::CATALOGS( <"Catalog"> )

returns physical attributes that are associated with catalogs that are accessible from the DBMS.

OLEDB::CHARACTER\_SETS( <"Catalog", "Schema", "Character-Set-Name"> )

returns the character sets that are specified in the catalog that a given user can access.

OLEDB::CHECK\_CONSTRAINTS(<"Catalog", "Schema", "Constraint-Name">)

returns check constraints that are specified in the catalog and that a given user owns.

OLEDB::COLLATIONS(<"Catalog", "Schema", "Collation-Name">)

returns the character collations that are specified in the catalog and that a given user can access.

OLEDB::COLUMN\_DOMAIN\_USAGE( <"Catalog", "Schema", "Domain-Name", "Column-Name"> )

returns the columns that are specified in the catalog, are dependent on a domain that is specified in the catalog, and a given user owns.

OLEDB::COLUMN\_PRIVILEGES( <"Catalog", "Schema", "Table-Name", "ColumnName", "Grantor", "Grantee"> )

returns the privileges on columns of tables that are specified in the catalog that a given user grants or can access.

OLEDB::COLUMNS( <"Catalog", "Schema", "Table-Name", "ColumnName"> )

returns the columns of tables that are specified in the catalogs that a given user can access.

OLEDB::CONSTRAINT\_COLUMN\_USAGE(<"Catalog", "Schema", "Table-Name", "ColumnName"> )

returns the columns that referential constraints, unique constraints, check constraints, and assertions use that are specified in the catalog and that a given user owns.

OLEDB::CONSTRAINT\_TABLE\_USAGE(<"Catalog", "Schema", "Table-Name"> )

returns the tables that referential constraints, unique constraints, check constraints, and assertions use that are specified in the catalog and that a given user owns.

OLEDB::FOREIGN\_KEYS(<"Primary-Key-Catalog", "Primary-Key-Schema", "Primary-Key-Table-Name", "Foreign-Key-Catalog", "Foreign-Key-Schema", "Foreign-Key-Table-Name">)  
     returns the foreign key columns that a given user specified in the catalog.

OLEDB::INDEXES( <"Catalog", "Schema", "Index-Name", "Type", "Table-Name"> )  
     returns the indexes that are specified in the catalog that a given user owns.

OLEDB::KEY\_COLUMN\_USAGE(<"Constraint-Catalog", "Constraint-Schema", "Constraint-Name", "Table-Catalog", "Table-Schema", "Table-Name", "Column-Name">)  
     returns the columns that are specified in the catalog and that a given user has constrained as keys.

OLEDB::PRIMARY\_KEYS(<"Catalog", "Schema", "Table-Name">)  
     returns the primary key columns that a given user specified in the catalog.

OLEDB::PROCEDURE\_COLUMNS(<"Catalog", "Schema", "Procedure-Name", "Column-Name">)  
     returns information about the columns of rowsets that procedures return.

OLEDB::PROCEDURE\_PARAMETERS(<"Catalog", "Schema", "Procedure-Name", "Parameter-Name">)  
     returns information about the parameters and return codes of the procedures.

OLEDB::PROCEDURES(<"Catalog", "Schema", "Procedure-Name", "Procedure-Type">)  
     returns procedures that are specified in the catalog that a given user owns.

OLEDB::PROVIDER\_INFO()  
     returns output that contains these columns: PROVIDER\_NAME, PROVIDER\_DESCRIPTION, and PROVIDER\_PROPERTIES. The PROVIDER\_PROPERTIES column contains a list of all properties that the provider supports. A semicolon (;) separates the properties. See ["Examples of Special OLE DB Queries" on page 1118](#).

OLEDB::PROVIDER\_TYPES(<"Data Type", "Best-Match">)  
     returns information about the base data types that the data provider supports.

OLEDB::REFERENTIAL\_CONSTRAINTS(<"Catalog", "Schema", "Constraint-Name">)  
     returns the referential constraints that are specified in the catalog that a given user owns.

OLEDB::SCHEMATA(<"Catalog", "Schema", "Owner">)  
     returns the schemas that a given user owns.

OLEDB::SQL\_LANGUAGES()  
     returns the conformance levels, options, and dialects that the SQL implementation processing data supports and that is specified in the catalog.

OLEDB::STATISTICS(<"Catalog", "Schema", "Table-Name">)  
     returns the statistics that is specified in the catalog that a given user owns.

OLEDB::TABLE\_CONSTRAINTS(<"Constraint-Catalog", "Constraint-Schema", "Constraint-Name", "Table-Catalog", "Table-Schema", "Table-Name", "Constraint-Type">)  
     returns the table constraints that is specified in the catalog that a given user owns.

OLEDB::TABLE\_PRIVILEGES(<"Catalog", "Schema", "Table-Name", "Grantor", "Grantee">)  
 returns the privileges on tables that are specified in the catalog that a given user grants or can access.

OLEDB::TABLES(<"Catalog", "Schema", "Table-Name", "Table-Type">)  
 returns the tables specified in the catalog that a given user grants and can access.

OLEDB::TRANSLATIONS(<"Catalog", "Schema", "Translation-Name">)  
 returns the character translations that are specified in the catalog and that are accessible to a given user.

OLEDB::USAGE\_PRIVILEGES(<"Catalog", "Schema", "Object-Name", "Object-Type", "Grantor", "Grantee">)  
 returns the USAGE privileges on objects that are specified in the catalog and that a given user grants or can access.

OLEDB::VIEW\_COLUMN\_USAGE(<"Catalog", "Schema", "View-Name">)  
 returns the columns on which viewed tables depend that are specified in the catalog and that a given user owns.

OLEDB::VIEW\_TABLE\_USAGE(<"Catalog", "Schema", "View-Name">)  
 returns the tables on which viewed tables depend that are specified in the catalog and that a given user owns.

OLEDB::VIEWS(<"Catalog", "Schema", "Table-Name">)  
 returns the viewed tables that are specified in the catalog and that a given user can access.

For a complete description of each rowset and the columns that are specified in each rowset, see the *Microsoft OLE DB Programmer's Reference*.

---

## Examples of Special OLE DB Queries

This example retrieves a rowset that displays all tables that the HRDEPT schema accesses:

```
proc sql;
  connect to oledb(provider=sqloledb properties=("User ID"=myusr1
  Password=mypwd1
  "Data Source"='dwtsrv1'));
  select * from connection to oledb
    (OLEDB::TABLES(,"HRDEPT"));
quit;
```

It uses the special query OLEDB::PROVIDER\_INFO() to produce this output:

```
proc sql;
  connect to oledb(provider=msdaora properties=("User ID"=myusr1
  Password=mypwd1
  "Data Source"="Oraserver"));
  select * from connection to oledb
    (OLEDB::PROVIDER_INFO());
quit;
```

**Output 31.1 Provider and Properties Output**

| PROVIDER_NAME | PROVIDER_DESCRIPTION                 | PROVIDER_PROPERTIES                                                                                        |
|---------------|--------------------------------------|------------------------------------------------------------------------------------------------------------|
| MSDAORA       | Microsoft OLE DB Provider for Oracle | Password;User ID;Data Source;Window Handle;Locale Identifier;OLE DB Services; Prompt; Extended Properties; |
| SampProv      | Microsoft OLE DB Sample Provider     | Data Source;Window Handle; Prompt;                                                                         |

You could then reference the output when automating a connection to the provider. For the previous result set, you could write this SAS/ACCESS LIBNAME statement:

```
libname mydblib oledb provider=msdaora
      props= ('Data Source'=OraServer 'User ID'=myusr1 'Password'=mypwd1);
```

---

## Temporary Table Support for OLE DB

SAS/ACCESS Interface to OLE DB supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

---

## Passing SAS Functions to OLE DB

SAS/ACCESS Interface to OLE DB passes the following SAS functions for OLE DB to DB2, Microsoft SQL Server, and Oracle for processing. Where the OLE DB function name differs from the SAS function name, the OLE DB name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|        |         |
|--------|---------|
| DAY    | SECOND  |
| HOUR   | WEEKDAY |
| MINUTE | YEAR    |
| MONTH  |         |

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to OLE DB. Due to incompatibility in date and time functions between OLE DB and SAS, OLE DB might not process them correctly. Check your results to determine whether these functions are working as expected.

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this

function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

|              |                |
|--------------|----------------|
| ABS          | LOWCASE        |
| ARCOS (ACOS) | MOD (see note) |
| ARSIN (ASIN) | QTR            |
| ATAN         | REPEAT         |
| Avg          | SIGN           |
| BYTE         | SIN            |
| CEIL         | SOUNDEX        |
| COMPRESS     | SQRT           |
| COS          | STRIP (TRIM)   |
| DATE         | SUBSTR         |
| DATEPART     | TAN            |
| DATETIME     | TIME           |
| EXP          | TIMEPART       |
| FLOOR        | TODAY          |
| INDEX        | TRANWRD        |
| LENGTH       | TRIMN          |
| LOG          | UPCASE         |
| LOG10        |                |

---

## Passing Joins to OLE DB

For a multiple libref join to pass to OLE DB, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- data source (DATASOURCE=)
- provider (PROVIDER=)
- qualifier (QUALIFIER=, if specified)
- provider string (PROVIDER\_STRING, if specified)
- path and file name (UDL\_FILE=, if specified)
- initialization string (INIT\_STRING=, if specified)
- read isolation level (READ\_ISOLATION\_LEVEL=, if specified)
- update isolation level (UPDATE\_ISOLATION\_LEVEL=, if specified)
- all properties (PROPERTIES=)
- prompt (PROMPT=, must *not* be specified)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Bulk Loading for OLE DB

The **BULKLOAD= LIBNAME** option calls the SQLOLEDB interface of IRowsetFastLoad so that you can efficiently insert rows of data into a Microsoft SQL Server database table as a unit. BCP= is an alias for this option.

---

**Note:** This functionality is available only when accessing Microsoft SQL Server data on Windows platforms using Microsoft SQL Server Version 7.0 or later.

---

As the OLE DB interface sends rows of data to the bulk-load facility, data is written to an input buffer. When you send all rows or when the buffer reaches a certain size (**DBCOMMIT=** determines this), all rows are inserted as a unit into the table and the data is committed to the table. You can also set **DBCOMMIT=** to commit rows after a specified number of rows are inserted.

If an error occurs, a message is written to the SAS log, and any rows that were inserted before the error are rolled back.

If you specify **BULKLOAD=YES** and the **PROVIDER=** option is specified, SAS/ACCESS Interface to OLE DB uses the specified provider. If you specify **BULKLOAD=YES** and **PROVIDER=** is not specified, the engine uses the **PROVIDER=SQLOLEDB** value.

If you specify **BULKLOAD=YES**, connections that are made through OLE DB Services or UDL files are not allowed.

---

## Locking in the OLE DB Interface

These LIBNAME and data set options let you control how the OLE DB interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134.

- **READ\_LOCK\_TYPE=** ROW | NOLOCK
- **UPDATE\_LOCK\_TYPE=** ROW | NOLOCK
- **READ\_ISOLATION\_LEVEL=** S | RR | RC | RU

The data provider specifies the default value. OLE DB supports the S, RR, RC, and RU isolation levels that are specified in this table.

*Table 31.3 Isolation Levels for OLE DB*

| Isolation Level  | Definition                                                         |
|------------------|--------------------------------------------------------------------|
| S (serializable) | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads. |

| Isolation Level       | Definition                                                                   |
|-----------------------|------------------------------------------------------------------------------|
| RR (repeatable Read)  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads. |
| RC (committed Read )  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads. |
| RU (uncommitted Read) | Allows dirty Reads, nonrepeatable Reads, and phantom Reads.                  |

Here is how the terms in the table are defined.

#### *Dirty reads*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it can see changes that are made by those concurrent transactions even before they commit.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

#### *Nonrepeatable reads*

If a transaction exhibits this phenomenon, it might read a row once. Then, if the same transaction attempts to read that row again, the row might have been changed or even deleted by another concurrent transaction. Therefore, the Read is not (necessarily) repeatable.

For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

#### *Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist, a phantom.

#### ■ `UPDATE_ISOLATION_LEVEL= S | RR | RC`

The default value is specified by the data provider. OLE DB supports the S, RR, and RC isolation levels defined in the preceding table. The RU isolation level is not allowed with this option.

---

# Accessing OLE DB for OLAP Data

---

## Overview

SAS/ACCESS Interface to OLE DB provides a facility for accessing OLE DB for OLAP data. You can specify a Multidimensional Expressions (MDX) statement through the SQL pass-through facility to access the data directly, or you can create an SQL view of the data. If your MDX statement specifies a data set with more than five axes (COLUMNS, ROWS, PAGES, SECTIONS, and CHAPTERS), SAS returns an error. See the Microsoft Data Access Components Software Developer's Kit for details about MDX syntax.

---

**Note:** This implementation provides Read-Only access to OLE DB for OLAP data. You cannot update or insert data with this facility.

---

---

## Using the SQL Pass-Through Facility with OLAP Data

---

### Overview

The main difference between normal OLE DB access using the SQL pass-through facility and the implementation for OLE DB for OLAP is the use of these additional identifiers to pass MDX statements to the OLE DB for OLAP data:

MDX::

identifies MDX statements that return a flattened data set from the multidimensional data.

MDX\_DESCRIBE::

identifies MDX statements that return detailed column information.

An MDX\_DESCRIBE:: identifier is used to obtain detailed information about each returned column. During the process of flattening multidimensional data, OLE DB for OLAP builds column names from each level of the given dimension. For example, for OLE DB for OLAP multidimensional data that contains CONTINENT, COUNTRY, REGION, and CITY dimensions, you could build a column with this name:

[NORTH AMERICA] . [USA] . [SOUTHEAST] . [ATLANTA]

This name cannot be used as a SAS variable name because it has more than 32 characters. For this reason, the SAS/ACCESS engine for OLE DB creates a column name based on a shortened description, in this case, ATLANTA. However, because

there could be an ATLANTA in some other combination of dimensions, you might need to know the complete OLE DB for OLAP column name. Using the MDX\_DESCRIBE:: identifier returns a SAS data set that contains the SAS name for the returned column and its corresponding OLE DB for OLAP column name:

| SASNAME   | MDX_UNIQUE_NAME                                     |
|-----------|-----------------------------------------------------|
| ATLANTA   | [NORTH AMERICA] . [USA] . [SOUTHEAST] . [ATLANTA]   |
| CHARLOTTE | [NORTH AMERICA] . [USA] . [SOUTHEAST] . [CHARLOTTE] |
| .         | .                                                   |
| .         | .                                                   |
| .         | .                                                   |

If two or more SASNAME values are identical, a number is appended to the end of the second and later instances of the name. For example, the values might be called ATLANTA, ATLANTA0, ATLANTA1, and so on. Also, depending on the value of the VALIDVARNAME= system option, invalid characters are converted to underscores in the SASNAME value.

---

## Syntax

This facility uses the following general syntax. For more information about SQL pass-through facility syntax, see “SQL Pass-Through Facility” on page 690.

```
PROC SQL <options>;
  CONNECT TO OLEDB (<options>);
  <non-SELECT SQL statement(s)>
  SELECT column-identifier(s) FROM CONNECTION TO OLEDB
    ( MDX:: | MDX_DESCRIBE:: <MDX statement>)
  <other SQL statement(s)>
;
```

---

## Examples

This code uses the SQL pass-through facility to pass an MDX statement to a Microsoft SQL Server Decision Support Services (DSS) Cube. The provider used is the Microsoft OLE DB for OLAP provider named MSOLAP.

```
proc sql noerrorstop;
  connect to oledb (provider=msolap prompt=yes);
  select * from connection to oledb
    ( MDX::select { [Measures].[Units Shipped],
      [Measures].[Units Ordered]} on columns,
      NON EMPTY [Store].[Store Name].members on rows
      from Warehouse );
```

See the Microsoft Data Access Components Software Developer's Kit for details about MDX syntax.

The CONNECT statement requests prompting for connection information, which facilitates the connection process (especially with provider properties). The MDX:: prefix identifies the statement within the parentheses that follows the MDX

statement syntax and is not an SQL statement that is specific to OLAP. Partial output from this query might look like this:

| Store  | Units Shipped | Units Ordered |
|--------|---------------|---------------|
| Store6 | 10,647        | 11,699        |
| Store7 | 24,850        | 26,223        |
| .      | .             | .             |
| .      | .             | .             |
| .      | .             | .             |

You can use the same MDX statement with the MDX\_DESCRIBE:: identifier to see the full description of each column:

```
proc sql noerrorstop;
connect to oledb (provider=msolap prompt=yes);
select * from connection to oledb
( MDX_DESCRIBE::select { [Measures]. [Units Shipped],
[Measures]. [Units Ordered] } on columns,
NON EMPTY [Store]. [Store Name].members on rows
from Warehouse );
```

The next example creates a view of the OLAP data, which is then accessed using the PRINT procedure:

```
proc sql noerrorstop;
connect to oledb(provider=msolap
      props=('data source'=sqlserverdb
             'user id'=myuserid password=mypassword));
create view work.myview as
select * from connection to oledb
( MDX::select { [MEASURES]. [Unit Sales] } on columns,
order(except([Promotion Media]. [Media Type].members,
{ [Promotion Media]. [Media Type]. [No Media]}),
[Measures]. [Unit Sales],DESC) on rows
from Sales )
;

proc print data=work.myview;
run;
```

In this example, full connection information is provided in the CONNECT statement, so the user is not prompted. The SQL view can be used in other PROC SQL statements, the DATA step, or in other procedures. However, you cannot modify (that is, insert, update, or delete a row in) the view's underlying multidimensional data.

## Naming Conventions for OLE DB

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Because OLE DB is an application programming interface (API), data source names for files, tables, and columns are determined at run time. Most SAS names can be up to 32 characters long. SAS/ACCESS Interface to OLE DB also supports file,

table, and column names up to 32 characters long. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a name results in identical names, SAS generates unique names by replacing the last character with a number. For more information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).)

---

## Data Types for OLE DB

---

### Overview

Each data source column in a table has a name and a data type. The data type tells the data source how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about OLE DB null and default values and data conversions.

---

## OLE DB Null Values

Many relational database management systems have a special value called NULL. A DBMS NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a DBMS NULL value, it interprets it as a SAS missing value.

In most relational databases, columns can be specified as NOT NULL so that they require data (they cannot contain NULL values). When a column is specified as NOT NULL, the DBMS does not add a row to the table unless the row has a value for that column. When creating a DBMS table with SAS/ACCESS, you can use the **DBNULL=** data set option to indicate whether NULL is a valid value for specified columns.

OLE DB mirrors the behavior of the underlying DBMS with regard to NULL values. See the documentation for your DBMS for information about how it handles NULL values.

For more information about how SAS handles NULL values, see [“Potential Result Set Differences When Processing Null Data” on page 43](#).

To control how the DBMS handles SAS missing character values, use the **NULCHAR=** and **NULCHARVAL=** data set options.

## Specifying the BOOL\_VAL Environment Variable

By default, the SAS/ACCESS LIBNAME engine imports YES (TRUE) into SAS as the numeric value 1. Set the environment variable with this statement.

```
/* To have the YES value imported into SAS as numeric value-1 */
OPTIONS SET=BOOL_VAL ASIS;

/* Reset to the default value */
OPTIONS SET=BOOL_VAL SAS;
```

---

**Note:** Support for the BOOL\_VAL environment variable was added for SAS 9.4.

---

## LIBNAME Statement Data Conversions

This table shows all data types and default SAS formats that SAS/ACCESS Interface to OLE DB supports. It does not explicitly specify the data types as they exist for each data source. It lists the types that each data source's data type might map to. For example, an INTEGER data type under DB2 might map to an OLE DB data type of DBTYPE\_I4. All data types are supported.

*Table 31.4 OLE DB Data Types and Default SAS Formats*

| OLE DB Data Type | Default SAS Format |
|------------------|--------------------|
| DBTYPE_R8        | none               |
| DBTYPE_R4        | none               |
| DBTYPE_I8        | none               |
| DBTYPE_UI8       | none               |
| DBTYPE_I4        | 11.                |
| DBTYPE_UI4       | 11.                |
| DBTYPE_I2        | 6.                 |
| DBTYPE_UI2       | 6.                 |
| DBTYPE_I1        | 4.                 |
| DBTYPE_UI1       | 4.                 |

| OLE DB Data Type   | Default SAS Format                                                                      |
|--------------------|-----------------------------------------------------------------------------------------|
| DBTYPE_BOOL        | 1.                                                                                      |
| DBTYPE_NUMERIC     | <i>m</i> or <i>w.d</i> or none, if <i>w</i> and <i>d</i> are not specified              |
| DBTYPE_DECIMAL     | <i>w</i> or <i>w.d</i> or none, if <i>w</i> and <i>d</i> are not specified              |
| DBTYPE_CY          | DOLLAR <i>w.2</i>                                                                       |
| DBTYPE_BYTES       | \$ <i>w.</i>                                                                            |
| DBTYPE_STR         | \$ <i>w.</i>                                                                            |
| DBTYPE_BSTR        | \$ <i>w.</i>                                                                            |
| DBTYPE_WSTR        | \$ <i>w.</i>                                                                            |
| DBTYPE_VARIANT     | \$ <i>w.</i>                                                                            |
| DBTYPE_DBDATE      | DATE9.                                                                                  |
| DBTYPE_DBTIME      | TIME8.                                                                                  |
| DBTYPE_DBTIMESTAMP | DATETIME <i>w.d</i> , where <i>w</i> depends on precision and <i>d</i> depends on scale |
| DBTYPE_DATE        |                                                                                         |
| DBTYPE_GUID        | \$38.                                                                                   |

The following table shows the default data types that SAS/ACCESS Interface to OLE DB uses when creating DBMS tables. SAS/ACCESS Interface to OLE DB lets you specify non-default data types by using the [DBTYPE=](#) data set option.

**Table 31.5 Default OLE DB Output Data Types**

| SAS Variable Format | Default OLE DB Data Type                                                           |
|---------------------|------------------------------------------------------------------------------------|
| <i>w.d</i>          | DBTYPE_R8 or DBTYPE_NUMERIC using <i>m.n</i> <sup>2</sup><br>if the DBMS allows it |
| \$ <i>w.</i>        | DBTYPE_STR using <i>n</i> <sup>1</sup>                                             |
| date formats        | DBTYPE_DBDATE                                                                      |
| time formats        | DBTYPE_DBTIME                                                                      |

| SAS Variable Format | Default OLE DB Data Type |
|---------------------|--------------------------|
| datetime formats    | DBTYPE_DBTIMESTAMP       |

1 *n* in OLE DB character data types is equivalent to *w* in SAS formats.  
2 *m* and *n* in OLE DB numeric data types are equivalent to *w* and *d* in SAS formats.

---

## Sample Programs for OLE DB

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to Oracle

---

|                                                               |      |
|---------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to Oracle</i> | 1132 |
| <i>Introduction to SAS/ACCESS Interface to Oracle</i>         | 1132 |
| <b>LIBNAME Statement for the Oracle Engine</b>                | 1133 |
| Overview                                                      | 1133 |
| Arguments                                                     | 1133 |
| Oracle LIBNAME Statement Examples                             | 1139 |
| <b>Data Set Options for Oracle</b>                            | 1139 |
| <b>SQL Pass-Through Facility Specifics for Oracle</b>         | 1143 |
| Key Information                                               | 1143 |
| CONNECT Statement Examples                                    | 1143 |
| <b>Autopartitioning Scheme for Oracle</b>                     | 1145 |
| Overview                                                      | 1145 |
| Partitioned Oracle Tables                                     | 1145 |
| Nonpartitioned Oracle Tables                                  | 1147 |
| Performance Summary                                           | 1148 |
| <b>ACCESS Procedure Specifics for Oracle</b>                  | 1148 |
| Overview                                                      | 1148 |
| Examples                                                      | 1149 |
| <b>DBLOAD Procedure Specifics for Oracle</b>                  | 1150 |
| Overview                                                      | 1150 |
| Examples                                                      | 1151 |
| <b>Maximizing Oracle Performance</b>                          | 1152 |
| <b>Temporary Table Support for Oracle</b>                     | 1152 |
| <b>Passing SAS Functions to Oracle</b>                        | 1152 |
| <b>Passing Joins to Oracle</b>                                | 1153 |
| <b>Sorting Data That Contains NULL Values</b>                 | 1154 |
| <b>Bulk Loading for Oracle</b>                                | 1154 |
| Overview                                                      | 1154 |
| Data Set Options with Bulk Loading                            | 1155 |
| Interactions with Other Options                               | 1155 |
| z/OS Specifics                                                | 1156 |

|                                              |      |
|----------------------------------------------|------|
| Examples .....                               | 1157 |
| <i>Locking in the Oracle Interface</i> ..... | 1158 |
| <i>Naming Conventions for Oracle</i> .....   | 1159 |
| <i>Data Types for Oracle</i> .....           | 1160 |
| Overview .....                               | 1160 |
| Supported Oracle Data Types .....            | 1160 |
| Example 1: Timestamp .....                   | 1161 |
| Example 2: Interval Year to Month .....      | 1163 |
| Example 3: Interval Day to Second .....      | 1163 |
| Default Data Types .....                     | 1164 |
| LIBNAME Statement Data Conversions .....     | 1165 |
| ACCESS Procedure Data Conversions .....      | 1167 |
| DBLOAD Procedure Data Conversions .....      | 1169 |
| <i>Sample Programs for Oracle</i> .....      | 1169 |

---

## System Requirements for SAS/ACCESS Interface to Oracle

You can find information about system requirements for SAS/ACCESS Interface to Oracle in the following locations:

- [System Requirements for SAS/ACCESS Interface to Oracle with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

## Introduction to SAS/ACCESS Interface to Oracle

For available SAS/ACCESS features, see [Oracle supported features on page 112](#). For more information about Oracle, see your Oracle documentation.

---

**Note:** Beginning with [SAS 9.4M8](#), SAS/ACCESS Interface to Oracle on the z/OS platform is no longer available. If you have an existing instance of SAS/ACCESS Interface to Oracle on z/OS and plan to upgrade to SAS 9.4M8 or later, then SAS/ACCESS recommends that you unconfigure and uninstall it. For more information, see [Unconfiguring and Uninstalling Retired Products](#).

---

SAS/ACCESS Interface to Oracle includes SAS Data Connector to Oracle. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- “Where to Specify Data Connector Options” in *SAS Cloud Analytic Services: User’s Guide*
- “Oracle Data Connector” in *SAS Cloud Analytic Services: User’s Guide*

# LIBNAME Statement for the Oracle Engine

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Oracle supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Oracle.

**LIBNAME** *libref oracle <connection-options> <LIBNAME-options>;*

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in *SAS Global Statements: Reference*.

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *oracle*

specifies the SAS/ACCESS engine name for the Oracle interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are defined.

**Note:** All of these connection options are valid when used in the CONNECT statement with the SQL procedure.

If you specify the appropriate system options or environment variables for Oracle, you can often omit the options from your LIBNAME statements. See your Oracle documentation for details.

**USER=<'>Oracle-user-name<'>**

specifies an optional Oracle user name. If the user name contains blanks or national characters, enclose it in quotation marks. If you omit an Oracle user name and password, the default Oracle user ID OPS\$*sysid* is used, if it is enabled.

Alias: **UID=**

Restriction: When you specify <'>Oracle-user-name<'> in the LIBNAME statement, SAS changes the user name to uppercase characters, even if the user name uses quotation marks.

**Table 32.1 How Oracle Treats the User Name**

| User Name Conditions                                                                | What to Know or Do                                                                                                                                     |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enclosed in single quotation marks:<br><b>'scott'</b>                               | The SAS/ACCESS Oracle engine ignores the single quotation marks, and Oracle converts characters to all uppercase (from <b>scott</b> to <b>SCOTT</b> ). |
| Contains lowercase or mixed-case characters:<br><b>scott</b><br><b>Scott</b>        | First enclose characters in double quotation marks, and then enclose everything in single quotation marks:<br><b>"scott"</b><br><b>"Scott"</b>         |
| Contains blanks or national characters:<br><b>scott smith</b>                       | Enclose characters in single or double quotation marks:<br><b>'scott smith'</b><br><b>"scott smith"</b>                                                |
| Contains a macro reference:<br><b>&amp;name</b>                                     | Enclose characters in double quotation marks:<br><b>"&amp;name"</b>                                                                                    |
| Contains one or more of these SQL*Loader special characters: = @ /<br><b>scott@</b> | Enclose all characters in single quotation marks, and then enclose everything in double quotation marks.<br><b>"'scott@'"</b>                          |

Required: If you specify USER=, you must also specify PASSWORD=.

**PASSWORD=<'>Oracle-password<'>**

specifies an optional Oracle password that is associated with the Oracle user name. If you omit it, the password for the default Oracle user ID OPS\$*sysid* is used if it is enabled. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: **ORAPW=, PASS=, PWD=, PW=**

Required: If you specify USER=, you must also specify PASSWORD=.

Important: The SQL\*Loader that the **BULKLOAD=** data set option calls takes user name and password as command-line arguments. This means that a user who runs the `ps` command while the SQL\*Loader is running could see the user credentials. To conceal the user name and password, be sure that the **BL\_PARFILE=** data set option is also specified.

Tip: If you do not wish to enter your Oracle password in uncoded text, see PROC PWENCODE in the *Base SAS Procedures Guide* for a method to encode it.

#### **PATH=<'>Oracle-database-specification<'>**

specifies the Oracle driver, node, and database. Aliases are required if you are using SQL\*Net Version 2.0 or later. In some operating environments, you can enter the information that is required by the PATH= statement before invoking SAS.

SAS/ACCESS uses the same Oracle path designation that you use to connect to Oracle directly. See your database administrator to determine the databases that have been set up in your operating environment and also the default values if you do not specify a database. To learn more about how to set up default connections to an Oracle database without specifying a value for the PATH environment variable, refer to the information about TWO\_TASK (on UNIX) or LOCAL (on Windows) in the environment variables section of your Oracle documentation.

#### **READBUFF=number-of-rows**

specifies the number of rows to retrieve from an Oracle table or view with each fetch. Using this option can improve the performance of any query to Oracle. By specifying the value of the READBUFF= option in your SAS programs, you can find the optimal number of rows for a given query on a given table.

Alias: BUFF= and BUFSIZE= are aliases for READBUFF=.

Default: 250 rows per fetch

Restriction: Although this value can be up to 2,147,483,647 rows per fetch, this depends on available memory. A practical limit for most applications is less, so the total maximum buffer size must not exceed 2GB.

#### **PRESERVE\_COMMENTS**

lets you pass additional information (called *hints*) to Oracle for processing. These hints might direct the Oracle query optimizer to choose the best processing method based on your hint.

You specify PRESERVE\_COMMENTS as an option in the CONNECT statement. You then specify the hints in the Oracle SQL query for the CONNECTION TO component. Hints are entered as comments in the SQL query and are passed to and processed by Oracle.

Restriction: This option is valid only in the CONNECTION statement, not in the LIBNAME statement. For the LIBNAME statement, you can use the PRESERVE\_COMMENTS= LIBNAME option.

#### **LIBNAME-options**

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The table below describes the LIBNAME options for SAS/ACCESS Interface to Oracle, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For more information, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 32.2** SAS/ACCESS LIBNAME Options for Oracle

| Option                               | Default Value                                                                              | Valid in CONNECT |
|--------------------------------------|--------------------------------------------------------------------------------------------|------------------|
| ACCESS=                              | none                                                                                       |                  |
| ADJUST_BYTE_SEMANTIC_COLUMN_LENGTHS= | conditional                                                                                | •                |
| ADJUST_NCHAR_COLUMN_LENGTHS=         | YES                                                                                        | •                |
| AUTHDOMAIN=                          | none                                                                                       | •                |
| BL_DEFAULT_DIR=                      | <database-name>                                                                            |                  |
| CONNECTION=                          | SHAREDREAD                                                                                 | •                |
| CONNECTION_GROUP=                    | none                                                                                       | •                |
| DB_LENGTH_SEMATICS_BYT=              | YES                                                                                        | •                |
| DB_OBJECTS=                          | (TABLES VIEWS)                                                                             |                  |
| DBCLIENT_ENCODING_FIXED=             | YES or NO, based on the SAS encoding                                                       | •                |
| DBCLIENT_MAX_BYTES=                  | matches the maximum number of bytes per single character of the SAS session encoding       | •                |
| DBCOMMIT=                            | 1000 (when inserting rows), 0 (when updating, deleting, or appending to an existing table) |                  |
| DBCONINIT=                           | none                                                                                       | •                |
| DBCONTERM=                           | none                                                                                       | •                |
| DBCREATE_TABLE_OPTS=                 | none                                                                                       |                  |
| DBGEN_NAME=                          | DBMS                                                                                       | •                |
| DBINDEX=                             | NO                                                                                         |                  |

| Option                   | Default Value                                                   | Valid in CONNECT |
|--------------------------|-----------------------------------------------------------------|------------------|
| DBLIBINIT=               | none                                                            |                  |
| DBLIBTERM=               | none                                                            |                  |
| DBLINK=                  | the local database                                              |                  |
| DBMAX_TEXT=              | 1024                                                            | •                |
| DBMSTEMP=                | NO                                                              |                  |
| DBNULLKEYS=              | YES                                                             |                  |
| DBNULLWHERE=             | YES                                                             |                  |
| DBPROMPT=                | NO                                                              | •                |
| DBSERVER_ENCODING_FIXED= | YES or NO, based on the Oracle server encoding                  | •                |
| DBSERVER_MAX_BYTES=      | usually 1                                                       | •                |
| DBSLICEPARM=             | THREADED_APPS,2                                                 |                  |
| DEFER=                   | NO                                                              | •                |
| DIRECT_EXE=              | none                                                            |                  |
| DIRECT_SQL=              | YES                                                             |                  |
| INSERTBUFF=              | 1 is the forced default when REREAD_EXPOSURE=YES; otherwise, 10 |                  |
| LOCKWAIT=                | YES                                                             |                  |
| MAX_STRING_SIZE=         | NONE                                                            |                  |
| MULTI_DATASRC_OPT=       | NONE                                                            |                  |
| POST_STMT_OPTS=          | none                                                            |                  |
| OR_BINARY_DOUBLE=        | YES                                                             | •                |
| OR_ENABLE_INTERRUPT=     | NO                                                              | •                |

| Option                  | Default Value                                          | Valid in CONNECT |
|-------------------------|--------------------------------------------------------|------------------|
| OR_UPD_NOWHERE=         | YES                                                    |                  |
| POST_DML_STMT_OPTS=     | none                                                   |                  |
| PRESERVE_COL_NAMES=     | NO                                                     |                  |
| PRESERVE_COMMENTS=      | NO                                                     | •                |
| PRESERVE_TAB_NAMES=     | NO                                                     |                  |
| READBUFF=               | 250                                                    | •                |
| READ_ISOLATION_LEVEL=   | see “Locking in the Oracle Interface”                  |                  |
| READ_LOCK_TYPE=         | NOLOCK                                                 | •                |
| REREAD_EXPOSURE=        | NO                                                     | •                |
| SCHEMA=                 | SAS accesses objects in the default and public schemas |                  |
| SHOW_SYNONYMS=          | NO                                                     |                  |
| SPOOL=                  | YES                                                    |                  |
| SQL_FUNCTIONS=          | none                                                   |                  |
| SQL_FUNCTIONS_COPY=     | none                                                   |                  |
| SQLGENERATION=          | none                                                   |                  |
| UPDATE_ISOLATION_LEVEL= | see “Locking in the Oracle Interface”                  |                  |
| UPDATE_LOCK_TYPE=       | NOLOCK                                                 | •                |
| UPDATEBUFF=             | 1                                                      |                  |
| UTILCONN_TRANSIENT=     | NO                                                     |                  |

---

## Oracle LIBNAME Statement Examples

This example uses default values for the connection options to make the connection. If you specify the appropriate system options or environment variables for Oracle, you can often omit the connection options from your LIBNAME statements. See your Oracle documentation for details.

```
libname myoralib oracle;
```

In this next example, the MYDBLIB libref uses the Oracle interface to connect to an Oracle database. SAS/ACCESS connection options are USER=, PASSWORD=, and PATH=. PATH= specifies an alias for the database specification, which SQL\*Net requires.

```
libname mydblib oracle user=myusr1 password=mypwd1 path=mysrv1;

proc print data=mydblib.employees;
  where dept='CSR010';
run;
```

This next example connects to Oracle using the CONNECT\_DATA= descriptor.

```
libname x oracle user=myusr1 pw=mypwd1
path=(DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP) (HOST = pinkfloyd) (PORT =
1521))
      )
      (CONNECT_DATA =
        (SID = alien )
      )
    ) "
  ;
```

---

## Data Set Options for Oracle

All SAS/ACCESS data set options in this table are supported for Oracle. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 32.3** SAS/ACCESS Data Set Options for Oracle

| Option      | Default Value                                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| BL_BADFILE= | creates a file in the temporary file directory as specified in the SAS UTILLOC= system option <sup>1</sup> or with the default file specifications. |

| Option                   | Default Value                                                                                                                                                                                                                                                                                                                     |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BL_API_BULKLOAD=         | NO                                                                                                                                                                                                                                                                                                                                |
| BL_CONTROL=              | creates a file in the temporary file directory as specified in the SAS UTILLOC= system option <sup>1</sup> or with the default file specifications.                                                                                                                                                                               |
| BL_DATAFILE=             | creates a file in the temporary file directory as specified in the SAS UTILLOC= system option <sup>1</sup> or with the default file specifications.                                                                                                                                                                               |
| BL_DEFAULT_DIR=          | <database-name>                                                                                                                                                                                                                                                                                                                   |
| BL_DELETE_DATAFILE=      | YES                                                                                                                                                                                                                                                                                                                               |
| BL_DELETE_ONLY_DATAFILE= | none                                                                                                                                                                                                                                                                                                                              |
| BL_DIRECT_PATH=          | YES                                                                                                                                                                                                                                                                                                                               |
| BL_DISCARDFILE=          | creates a file in the temporary file directory as specified in the SAS UTILLOC= system option <sup>1</sup> or with the default file specifications.                                                                                                                                                                               |
| BL_INDEX_OPTIONS=        | none                                                                                                                                                                                                                                                                                                                              |
| BL_LOAD_METHOD=          | When loading an empty table, the default value is INSERT. When loading a table that contains data, the default value is APPEND.                                                                                                                                                                                                   |
| BL_LOG=                  | If a log file does not already exist, it creates a file in the temporary file directory as specified in the SAS UTILLOC= system option <sup>1</sup> or with the default file specifications. If a log file does already exist, the Oracle bulk loader reuses the file, replacing the contents with information from the new load. |
| BL_OPTIONS=              | ERRORS=1000000                                                                                                                                                                                                                                                                                                                    |
| BL_PARFILE=              | none                                                                                                                                                                                                                                                                                                                              |
| BL_PRESERVE_BLANKS=      | NO                                                                                                                                                                                                                                                                                                                                |

| <b>Option</b>                 | <b>Default Value</b>        |
|-------------------------------|-----------------------------|
| BL_RECOVERABLE=               | YES                         |
| BL_RETURN_WARNINGS_AS_ERRORS= | NO                          |
| BL_SUPPRESS_NULLIF=           | NO                          |
| BL_USE_PIPE=                  | NO                          |
| BULKLOAD=                     | NO                          |
| DBCOMMIT=                     | LIBNAME option value        |
| DB_ONE_CONNECT_PER_THREAD=    | YES                         |
| DBCONDITION=                  | none                        |
| DBCREATE_TABLE_OPTS=          | LIBNAME option value        |
| DBFORCE=                      | NO                          |
| DBGEN_NAME=                   | DBMS                        |
| DBINDEX=                      | LIBNAME option value        |
| DBKEY=                        | none                        |
| DBLABEL=                      | NO                          |
| DBLINK=                       | LIBNAME option value        |
| DBLARGETABLE=                 | none                        |
| DBMAX_TEXT=                   | 1024                        |
| DBNULL=                       | YES                         |
| DBNULLKEYS=                   | LIBNAME option value        |
| DBNULLWHERE=                  | LIBNAME option value        |
| DBPROMPT=                     | LIBNAME option value        |
| DBSASLABEL=                   | COMPAT                      |
| DBSASTYPE=                    | see “Data Types for Oracle” |
| DBSLICE=                      | none                        |

| Option                  | Default Value                            |
|-------------------------|------------------------------------------|
| DBSLICEPARM=            | THREADED_APPS,2                          |
| DBTYPE=                 | see “LIBNAME Statement Data Conversions” |
| ERRLIMIT=               | 1                                        |
| INSERTBUFF=             | LIBNAME option value                     |
| NULLCHAR=               | SAS                                      |
| NULLCHARVAL=            | a blank character                        |
| OR_IDENTITY_COLS=       | none                                     |
| OR_PARTITION=           | an Oracle table partition name           |
| OR_UPD_NOWHERE=         | LIBNAME option value                     |
| ORHINTS=                | no hints                                 |
| POST_DML_STMT_OPTS=     | none                                     |
| POST_STMT_OPTS=         | none                                     |
| POST_TABLE_OPTS=        | none                                     |
| PRE_STMT_OPTS=          | none                                     |
| PRE_TABLE_OPTS=         | none                                     |
| PRESERVE_COL_NAMES=     | LIBNAME option value                     |
| READ_ISOLATION_LEVEL=   | LIBNAME option value                     |
| READ_LOCK_TYPE=         | LIBNAME option value                     |
| READBUFF=               | LIBNAME option value                     |
| SASDATEFORMAT=          | DATETIME20.0                             |
| SCHEMA=                 | LIBNAME option value                     |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value                     |
| UPDATE_LOCK_TYPE=       | LIBNAME option value                     |

| Option      | Default Value        |
|-------------|----------------------|
| UPDATEBUFF= | LIBNAME option value |

**1** For details, see [SAS System Options: Reference](#).

# SQL Pass-Through Facility Specifics for Oracle

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Oracle interface.

- The *dbms-name* is `oracle`.
- The CONNECT statement is optional. If you omit it, an implicit connection is made with your OPS\$*sysid*, if it is enabled. When you omit a CONNECT statement, an implicit connection is performed when the first EXECUTE statement or CONNECTION TO component is passed to Oracle. In this case you must use the default DBMS name `oracle`.
- The Oracle interface can connect to multiple databases (both local and remote) and to multiple user IDs. If you use multiple simultaneous connections, you must use an *alias* option to identify each connection. If you do not specify an alias, the default alias, `oracle`, is used.
- Here are the *database-connection-options* for the CONNECT statement.
  - [PASSWORD=](#)
  - [PATH=](#)
  - [USER=](#)
  - [READBUFF=](#)
  - [PRESERVE\\_COMMENTS=](#)

## CONNECT Statement Examples

This example uses the alias DBCON for the DBMS connection. The connection alias is optional.

```
proc sql;
```

```

connect to oracle as dbcon
  (user=myusr1 password=mypwd1 bufsize=100
   path='mysrv1');
quit;

```

This next example connects to Oracle and sends it two EXECUTE statements to process.

```

proc sql;
  connect to oracle (user=myusr1 password=mypwd1);
  execute (create view whotookorders as
    select ordernum, takenby,
           firstname, lastname, phone
      from orders, employees
     where orders.takenby=employees.empid)
  by oracle;
  execute (grant select on whotookorders
            to myusr1) by oracle;
  disconnect from oracle;
quit;

```

As shown in highlighted text, this example performs a query on the CUSTOMERS Oracle table:

```

proc sql;
  connect to oracle (user=myusr1 password=mypwd1);
  select *
    from connection to oracle
      (select * from customers
       where customer like '1%');
  disconnect from oracle;
quit;

```

In this example, the PRESERVE\_COMMENTS option is specified after the USER= and PASSWORD= options. The Oracle SQL query is enclosed in the required parentheses. The SQL INDEX command identifies the index for the Oracle query optimizer to use to process the query. Multiple hints are separated with blanks.

```

proc sql;
  connect to oracle as mycon(user=myusr1
                                password=mypwd1 preserve_comments);
  select *
    from connection to mycon
      (select /* +indx(empid) all_rows */
              count(*) from employees);
quit;

```

Hints are not preserved in this next example, which uses an older style of syntax:

```

execute ( delete /*+ FIRST_ROWS */ from test2 where num2=1)
        by &db

```

Using the new syntax, hints are preserved in this example:

```

execute by &db
  ( delete /*+ FIRST_ROWS */ from test2 where num2=2);

```

# Autopartitioning Scheme for Oracle

## Overview

Without user-specified partitioning from the DBSLICE= option, SAS/ACCESS Interface to Oracle tries to use its own partitioning techniques. The technique that it chooses depends on whether the table is physically partitioned on the Oracle server.

For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

---

**Note:** Threaded Reads for the Oracle engine on z/OS are not supported.

---

## Partitioned Oracle Tables

If you are working with a partitioned Oracle table, it is recommended that you let the Oracle engine partition the table for you. The Oracle engine gathers all partition information needed to perform a threaded Read on the table.

A partitioned Oracle table is a good candidate for a threaded Read because each partition in the table can be read in parallel with little contention for disk resources. If the Oracle engine determines that the table is partitioned, it makes the same number of connections to the server as there are partitions. This is true as long as the maximum number of threads that are allowed is higher than the number of partitions. Each connection retrieves rows from a single partition.

If the value of the maximum number of allowed threads is less than the number of partitions on the table, a single connection reads multiple partitions. Each connection retrieves rows from a single partition or multiple partitions. However, you can use the DB\_ONE\_CONNECT\_PER\_THREAD= data set option so that there is only one connection per thread.

This example shows how to do this. First, create the SALES table in Oracle.

```
CREATE TABLE SALES (acct_no NUMBER(5),
    acct_name CHAR(30), amount_of_sale NUMBER(6), qtr_no INTEGER)
PARTITION BY RANGE (qtr_no)
(PARTITION sales1 VALUES LESS THAN (2) TABLESPACE ts0,
PARTITION sales2 VALUES LESS THAN (2) TABLESPACE ts1,
PARTITION sales3 VALUES LESS THAN (2) TABLESPACE ts2,
PARTITION sales4 VALUES LESS THAN (2) TABLESPACE ts3)
```

Performing a threaded Read on this table with the following code, SAS makes four separate connections to the Oracle server and each connection reads from each partition. Turning on SASTRACE= shows the SQL that is generated for each connection.

```

libname x oracle user=myusr1 path=mysrv1;
data new;
set x.SALES (DBSLICEPARM=(ALL,10));
run;

ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES2)
ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES3)
ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES1)
ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES4)

```

Using the following code, SAS instead makes two separate connections to the Oracle server and each connection reads from two different partitions.

```

libname x oracle user=myusr1 path=mysrv1;
data new;
set x.SALES (DBSLICEPARM=(ALL,2));
run;

ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES2) UNION ALL SELECT "ACCT_NO", "ACCT_NAME",
"AMOUNT_OF_SALE",
"QTR_NO" FROM SALES partition (SALES3)
ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES1) UNION ALL SELECT "ACCT_NO", "ACCT_NAME",
"AMOUNT_OF_SALE",
"QTR_NO" FROM SALES partition (SALES4)

```

Using **DB\_ONE\_CONNECT\_PER\_THREAD=NO**, however, you can override the default behavior of limiting the number of connections to the number of threads. As shown below, SAS makes four separate connections to the Oracle server and each connection reads from each of the partition.

```

libname x oracle user=myusr1 path=mysrv1;
data new;
set x.SALES (DBSLICEPARM=(ALL,2)  DB_ONE_CONNECT_PER_THREAD=NO );
run;

ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES2)
ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES3)
ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES
partition (SALES1)
ORACLE: SELECT "ACCT_NO", "ACCT_NAME", "AMOUNT_OF_SALE", "QTR_NO" FROM
SALES

```

```
partition (SALES4)
```

The second parameter of the DBSLICEPARM= LIBNAME option determines the number of threads to read the table in parallel. The number of partitions on the table, the maximum number of allowed threads, and the value of DB\_ONE\_CONNECT\_PER\_THREAD= determine the number of connections to the Oracle server for retrieving rows from the table.

## Nonpartitioned Oracle Tables

If the table is not partitioned and the DBSLICE= option is not specified, Oracle resorts to the MOD function. (See “[Autopartitioning Techniques in SAS/ACCESS](#)” on [page 76](#). ) With this technique, the engine makes  $N$  connections, and each connection retrieves rows based on a WHERE clause as shown below.

```
WHERE ABS (MOD (ModColumn,N) ) =R
```

- ModColumn is a column in the table of type integer and is not used in any user specified WHERE clauses. (The engine selects this column. If you do not think this is the ideal partitioning column, you can use the DBSLICE= data set option to override this default behavior.)
- R varies from 0 to  $(N-1)$  for each of the  $N$  WHERE clauses.
- $N$  defaults to 2, and  $N$  can be overridden with the second parameter in the DBSLICEPARM= data set option.

The Oracle engine selects the ModColumn to use in this technique. Any numeric column with zero scale value can qualify as the ModColumn. However, if a primary key column is present, it is preferred over all others. Generally, values in the primary key column are in a serial order and yield an equal number of rows for each connection. This example illustrates the point:

```
create table employee (empno number(10) primary key,
                      empname varchar2(20), hiredate date,
                      salary number(8,2), gender char(1));
```

Performing a threaded Read on this table causes Oracle to make two separate connections to the Oracle server. SAS tracing shows the SQL generated for each connection:

```
data new;
set x.EMPLOYEE(DBSLICPARM=ALL);
run;
ORACLE: SELECT "EMPNO", "EMPNAME", "HIREDATE", "SALARY", "GENDER"
FROM EMPLOYEE WHERE ABS(MOD("EMPNO",2))=0
ORACLE: SELECT "EMPNO", "EMPNAME", "HIREDATE", "SALARY", "GENDER"
FROM EMPLOYEE WHERE ABS(MOD("EMPNO",2))=1
```

EMPNO, the primary key, is selected as the MOD column.

The success of MOD depends on the distribution of the values within the selected ModColumn and the value of  $N$ . Ideally, the rows should be distributed evenly among the threads.

You can alter the  $N$  value by changing the second parameter of DBSLICEPARM= LIBNAME option.

## Performance Summary

There are times that you might not see an improvement in performance with the MOD technique. It is possible that the engine might not be able to find a column that qualifies as a good MOD column. In these situations, you can explicitly specify DBSLICE= data set option to force a threaded Read and improve performance.

It is a good policy to let the engine autopartition and intervene with DBSLICE= only when necessary.

## ACCESS Procedure Specifics for Oracle

### Overview

For general information about this feature, see [Appendix 1, “ACCESS Procedure,” on page 1421](#). Oracle examples are available.

The Oracle interface supports all ACCESS procedure statements in line and batch modes. See [“About ACCESS Procedure Statements” on page 1422](#).

Here are the ACCESS procedure specifics for Oracle.

- The PROC ACCESS step DBMS= value is Oracle.
- Here are the *database-description-statements* that PROC ACCESS uses:

**USER=<'>Oracle-user-name<'>**

specifies an optional Oracle user name. If you omit an Oracle password and user name, the default Oracle user ID OPS\$*sysid* is used if it is enabled. If you specify USER=, you must also specify ORAPW=.

Restriction: PROC ACCESS does not change the value that you specify in USER= to uppercase if you enclose it within single quotation marks. This is a behavior differs from how it is handled in the LIBNAME statement.

**ORAPW=<'>Oracle-password<'>**

specifies an optional Oracle password that is associated with the Oracle user name. If you omit ORAPW=, the password for the default Oracle user ID OPS\$*sysid* is used, if it is enabled. If you specify ORAPW=, you must also specify USER=.

Restriction: PROC ACCESS does not change the value that you specify in ORAPW= to uppercase if you enclose it within single quotation marks. This is a behavior differs from how this is handled in the LIBNAME statement.

**PATH=<'>Oracle-database-specification<'>**

specifies the Oracle driver, node, and database. Aliases are required if you are using SQL\*Net Version 2.0 or later. In some operating environments, you can enter the information that is required by the PATH= statement before invoking SAS.

SAS/ACCESS uses the same Oracle path designation that you use to connect to Oracle directly. See your database administrator to determine the databases that have been set up in your operating environment and also the default values if you do not specify a database. To learn more about how to set up default connections to an Oracle database without specifying a value for the PATH environment variable, see the information about TWO\_TASK (on UNIX) or LOCAL (on Windows) in the environment variables section of your Oracle documentation.

- Here is the PROC ACCESS step TABLE= statement:

**TABLE= <'><Oracle-table-name><'>;**  
 specifies the name of the Oracle table or Oracle view on which the access descriptor is based. This statement is required. The *Oracle-table-name* option can be up to 30 characters long and must be a valid Oracle table name. If the table name contains blanks or national characters, enclose it in quotation marks.

## Examples

This example creates an access descriptor and a view descriptor based on Oracle data.

```
options linesize=80;

libname adlib 'SAS-library';
libname vlib 'SAS-library';

proc access dbms=oracle;
/* Create an access descriptor */
create adlib.customer.access;
user=myusr1;
orapw=mypwd1;
table=customers;
path='mysrv1';
assign=yes;
rename customer=custnum;
format firstorder date9. ;
list all;

/* Create a view descriptor */
create vlib.usacust.view;
select customer state zipcode name firstorder;
subset where customer like '1%';
run;
```

This next example creates another view descriptor that is based on the ADLIB.CUSTOMER access descriptor. You can then print the view.

```
/* Create the socust view */
proc access dbms=oracle accdesc=adlib.customer;
create vlib.socust.view;
select customer state name contact;
subset where state in ('NC', 'VA', 'TX');
run;
```

```

/* Print the socust view */
proc print data=vlib.socust;
title 'Customers in Southern States';
run;

```

# DBLOAD Procedure Specifics for Oracle

## Overview

For general information about this feature, see [Appendix 3, “DBLOAD Procedure,” on page 1455](#). Oracle examples are available.

The Oracle interface supports all DBLOAD procedure statements. See [“About DBLOAD Procedure Statements” on page 1456](#).

Here are the DBLOAD procedure specifics for Oracle.

- The PROC DBLOAD step DBMS= value is Oracle.
- Here are the *database-description-statements* that PROC DBLOAD uses:

**USER=<'>Oracle-user-name<'>**

specifies an optional Oracle user name. If you omit an Oracle password and user name, the default Oracle user ID OPS\$*sysid* is used if it is enabled. If you specify USER=, you must also specify ORAPW=.

**ORAPW= <'>Oracle-password<'>**

specifies an optional Oracle password that is associated with the Oracle user name. If you omit ORAPW=, the password for the default Oracle user ID OPS\$*sysid* is used, if it is enabled. If you specify ORAPW=, you must also specify USER=.

**PATH=<'>Oracle-database-specification<'>**

specifies the Oracle driver, node, and database. Aliases are required if you are using SQL\*Net Version 2.0 or later. In some operating environments, you can enter the information that is required by the PATH= statement before invoking SAS.

SAS/ACCESS uses the same Oracle path designation that you use to connect to Oracle directly. See your database administrator to determine the databases that have been set up in your operating environment and also the default values if you do not specify a database. To learn more about how to set up default connections to an Oracle database without specifying a value for the PATH environment variable, see the information about TWO\_TASK (on UNIX) or LOCAL (on Windows) in the environment variables section of your Oracle documentation.

**TABLESPACE= <'>Oracle-tablespace-name<'>;**

specifies the name of the Oracle tablespace where you want to store the new table. The *Oracle-tablespace-name* option can be up to 18 characters long and must be a valid Oracle tablespace name. If the name contains blanks or national characters, enclose the entire name in quotation marks.

If TABLESPACE= is omitted, the table is created in your default tablespace that is specified by the Oracle database administrator at your site.

- Here is the PROC DBLOAD step TABLE= statement:

**TABLE= <'><Oracle-table-name><'>;**  
 specifies the name of the Oracle table or Oracle view on which the access descriptor is based. This statement is required. The *Oracle-table-name* option can be up to 30 characters long and must be a valid Oracle table name. If the table name contains blanks or national characters, enclose the name in quotation marks.

## Examples

This example creates a new Oracle table, EXCHANGE, from the DLIB.RATEOFEX data file. Based on the new table, an access descriptor, ADLIB.EXCHANGE, is also created. The PATH= statement uses an alias to connect to a remote Oracle database. The SQL statement in the second DBLOAD procedure sends an SQL GRANT statement to Oracle. You must be granted Oracle privileges to create new Oracle tables or to grant privileges to other users. The SQL statement is in a separate procedure because you cannot create a DBMS table and reference it within the same DBLOAD step. The new table is not created until the RUN statement is processed at the end of the first DBLOAD step.

```
libname adlib 'SAS-library';
libname dlib 'SAS-library';

proc dbload dbms=oracle data=dlib.rateofex;
  user=myusr1;
  orapw=mypwd1;
  path='mysrv1';
  table=exchange;
  accdesc=adlib.exchange;
  rename fgnindol=fgnindolar 4=dolrsinfgn;
  nulls updated=n fgnindol=n 4=n country=n;
  load;
run;

proc dbload dbms=oracle;
  user=myusr1;
  orapw=mypwd1;
  path='mysrv1';
  sql grant select on myusr1.exchange to pham;
run;
```

This example uses the APPEND option to append rows from the INVDATA data set. This data set was created previously to an existing Oracle table named INVOICE.

```
proc dbload dbms=oracle data=invdata append;
  user=myusr1;
  orapw=mypwd1;
  path='mysrv1';
  table=invoice;
  load;
run;
```

---

## Maximizing Oracle Performance

You can take several measures to optimize performance when using SAS/ACCESS Interface to Oracle. For general information about improving performance when using SAS/ACCESS engines, see [Chapter 5, “Performance Considerations,” on page 47](#).

SAS/ACCESS Interface to Oracle has several options that you can use to further improve performance.

- For tips on multi-row processing, see these LIBNAME options: [INSERTBUFF](#), [READBUFF](#), and [UPDATEBUFF](#).
- For instructions on using the Oracle SQL\*Loader to increase performance when loading rows of data into Oracle tables, see [“Passing Functions to the DBMS Using PROC SQL” on page 58](#).

If you choose the transactional inserting of rows (specify BULKLOAD=NO), you can improve performance by inserting multiple rows at a time. This performance enhancement is comparable to using the Oracle SQL\*Loader Conventional Path Load. For more information about inserting multiple rows, see the INSERTBUFF= option.

---

## Temporary Table Support for Oracle

SAS/ACCESS Interface to Oracle supports temporary tables. For more information, see [“Temporary Table Support for SAS/ACCESS” on page 51](#).

---

## Passing SAS Functions to Oracle

SAS/ACCESS Interface to Oracle passes the following SAS functions to Oracle for processing. Where the Oracle function name differs from the SAS function name, the Oracle name appears in parentheses. For more information, see [“Passing Functions to the DBMS Using PROC SQL” on page 58](#).

|              |        |
|--------------|--------|
| ABS          | MAX    |
| ARCOS (ACOS) | MIN    |
| ARSIN (ASIN) | MINUTE |
| ATAN         | SECOND |
| ATAN2        | SIGN   |
| AVG          | SIN    |

|                    |                      |
|--------------------|----------------------|
| BAND               | SINH                 |
| CEIL               | SOUNDEX              |
| COS                | SQRT                 |
| COSH               | STD (STDDEV_SAMP)    |
| COUNT              | STRIP (TRIM)         |
| DATETIME (SYSDATE) | SUM                  |
| EXP                | TAN                  |
| FLOOR              | TANH                 |
| HOUR               | TRANSLATE (see note) |
| LOG                | TRIMN (RTRIM)        |
| LOG10              | UPCASE (UPPER)       |
| LOG2               | VAR (VAR_SAMP)       |
| LOWCASE (LCASE)    |                      |

**Note:** If more than three arguments are provided for the TRANSLATE function, then TRANSLATE is performed by SAS and the function is not passed to the database.

When the Oracle server is 9i or above, these additional functions are also passed.

|               |                 |
|---------------|-----------------|
| COALESCE      | MONTH (EXTRACT) |
| DAY (EXTRACT) | YEAR (EXTRACT)  |

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when SQL\_FUNCTIONS=ALL can the SAS/ACCESS engine also pass these SAS SQL functions to Oracle. Due to incompatibility in date and time functions between Oracle and SAS, Oracle might not process them correctly. Check your results to determine whether these functions are working as expected.

|                        |                         |
|------------------------|-------------------------|
| DATE (TRUNC(SYSDATE))* | SUBSTR                  |
| DATEPART (TRUNC)*      | TODAY (TRUNC(SYSDATE))* |
| INDEX (INSTR)          | TRANWRD (REPLACE)       |
| LENGTH                 | TRIM (TRIMN)            |
| MOD                    |                         |

\*Only in WHERE or HAVING clauses.

## Passing Joins to Oracle

Before a join across more than one libref can pass to Oracle, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- path (PATH=)

For all queries that use an ON clause, both operands in the ON clause must reference a column name. Literal operands cannot be passed down to the DBMS. Because literal operands are not supported, all ON clauses are transformed into WHERE clauses before passing joins down to the DBMS. This can result in a query not being passed down if it contains additional WHERE clauses or if it contains complex join conditions.

Oracle supports nonstandard outer-join syntax. Outer joins between two or more tables can be passed down to the DBMS with these restrictions:

- Full outer joins are not supported.
- Only a comparison operator is allowed in an ON clause.

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

## Sorting Data That Contains NULL Values

Some DBMSs, including Oracle, place NULL values at the end of a sorted list. SAS normally places NULL values at the beginning of a sorted list. In SAS, to use BY-group processing, SAS expects that values for a BY group are already sorted before it performs BY-group processing. If SAS determines that values are not sorted, then BY-group processing stops. Therefore, if NULL values are not at the beginning of a sorted list, SAS determines that the values are not sorted correctly.

If you use PROC SQL with an ORDER BY statement to sort your data, then the SQL code is generated so that NULL values are placed at the beginning of the sorted list. In this way, SAS can perform any BY-group processing without stopping.

For more information, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43 and “[Sorting DBMS Data](#)” on page 49.

## Bulk Loading for Oracle

### Overview

SAS/ACCESS Interface to Oracle can call the Oracle SQL\*Loader (SQLLDR) when you specify **BULKLOAD=YES**. The Oracle bulk loader provides superior load performance, so you can rapidly move data from a SAS file into an Oracle table.

**Note:** To preserve SQL\*Loader special characters (= @ /) in the user name (**USER=**) in the LIBNAME statement, you must enclose all characters in double quotation marks and then enclose everything in single quotation marks (for example, ' "scott@" ').

---

## Data Set Options with Bulk Loading

Here are the Oracle bulk-load data set options. For detailed information about these options, see [Data Set Options for Relational Databases on page 370](#).

- [BL\\_BADFILE=](#)
- [BL\\_CONTROL=](#)
- [BL\\_DATAFILE=](#)
- [BL\\_DEFAULT\\_DIR=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DIRECT\\_PATH=](#)
- [BL\\_DISCARDFILE=](#)
- [BL\\_INDEX\\_OPTIONS=](#)
- [BL\\_LOAD\\_METHOD=](#)
- [BL\\_LOG=](#)
- [BL\\_OPTIONS=](#)
- [BL\\_PARFILE=](#)
- [BL\\_PRESERVE\\_BLANKS=](#)
- [BL\\_RECOVERABLE=](#)
- [BL\\_RETURN\\_WARNINGS\\_AS\\_ERRORS=](#)
- [BL\\_SUPPRESS\\_NULLIF=](#)
- [BL\\_USE\\_PIPE=](#)
- [BULKLOAD=](#)

**BULKLOAD=** calls the Oracle bulk loader so that the Oracle engine can move data from a SAS file into an Oracle table using SQL\*Loader (SQLLDR).

---

**Note:** SQL\*Loader direct-path load has a number of limitations. See your Oracle utilities documentation for details, including tips to boost performance.

---

---

**Note:** You can also view the SQL\*Loader log file instead of the SAS log for information about the load when you use bulk loading.

---

---

## Interactions with Other Options

When **BULKLOAD=YES**, these statements are true:

- The [DBCOMMIT=](#), [DBFORCE=](#), [ERRLIMIT=](#), and [INSERTBUFF=](#) options are ignored.

- If **NULLCHAR=SAS**, and the **NULCHARVAL** value is blank, the SQL\*Loader attempts to insert a NULL instead of a NULCHARVAL value.
  - If **NULLCHAR=NO**, and the **NULCHARVAL** value is blank, the SQL\*Loader attempts to insert a NULL even if the DBMS does not allow NULL.
- To avoid this result, set **BL\_PRESERVE\_BLANKS=YES** or set **NULCHARVAL** to a non-blank value and then replace the non-blank value with blanks after processing, if necessary.

## z/OS Specifics

When you use bulk loading in the z/OS operating environment, the files that the SQL\*Loader uses must conform to z/OS data set standards. The data sets can be either sequential data sets or partitioned data sets. Each file name that is supplied to the SQL\*Loader are subject to extension and FNA processing.

If you do not specify file names using data set options, default names in the form of *userid.SAS.data-set-extension* apply. The *userid* is the TSO prefix when running under TSO, and it is the PROFILE PREFIX in batch. The *data-set-extensions* are:

- BAD for the bad file
- CTL for the control file
- DAT for the data file
- DSC for the discard file
- LOG for the log file

If you want to specify file names using data set options, you must use one of these forms:

- */DD/ddname*
- */DD/ddname(membername)*
- *Name*

For detailed information about these forms, see the SQL\*Loader chapter in the Oracle user's guide for z/OS.

The Oracle engine runs the SQL\*Loader by issuing a host-system command from within your SAS session. The data set where the SQLldr executable file resides must be available to your TSO session or allocated to your batch job. Check with your system administrator if you do not know the name or availability of the data set that contains the SQLldr executable file.

By default, on z/OS the bad file and discard file are not created in the same format as the data file. This makes it difficult to load the contents of these files after making corrections. See the section on SQL\*Loader file attributes in the SQL\*Loader section in the Oracle user's guide for z/OS for information about overcoming this limitation.

---

## Examples

This example shows how you can use a SAS data set, SASFLT.FLT98, to create and load an Oracle table, FLIGHTS98.

```
libname sasflt 'SAS-data-library';
libname mydblib oracle server=mysrv1_users
    db=users user=myusr1 password=mypwd1 path=mysrv1;

proc sql;
create table net_air.flights98
(BULKLOAD=YES
BL_DATAFILE='c:\temp\oracle\data.dat'
BL_USE_PIPE=NO
BL_DELETE_DATAFILE=yes
as select * from sasflt.flt98;
quit;
```

This next example shows how you can append the SAS data set, SASFLT.FLT98, to the existing Oracle table ALLFLIGHTS. The BL\_USE\_PIPE=NO option forces SAS/ACCESS Interface to Oracle to write data to a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the load has completed.

```
proc append base=new_air.flights98
(BULKLOAD=YES
BL_DATAFILE='c:\temp\oracle\data.dat'
BL_USE_PIPE=NO
BL_DELETE_DATAFILE=no
data=sasflt.flt98;
run;
```

This example shows you how to create and use a SAS data set to create and load to a large Oracle table, FLIGHTS98. This load uses the SQL\*Loader direct path method because you specified BULKLOAD=YES. BL\_OPTIONS= passes the specified SQL\*Loader options to SQL\*Loader when it is invoked. In this example, you can use the ERRORS= option to have up to 899 errors in the load before it terminates. Also, the LOAD= option loads the first 5,000 rows of the input data set, SASFLT.FLT98.

```
options yearcutoff=1940; /* included for Year 2000 compliance */

libname sasflt 'SAS-library';
libname ora_air oracle user=myusr1 password=mypwd1
    path='mysrv1_flt' schema=statsdiv;

data sasflt.flt98;
    input flight $3. +5 dates date7. +3 depart time5. +2 orig $3.
        +3 dest $3. +7 miles +6 boarded +6 capacity;
    format dates date9. depart time5.;
    informat dates date7. depart time5. ;
    datalines;
114      01JAN98     7:10   LGA    LAX          2475      172      210
202      01JAN98     10:43   LGA    ORD          740       151      210
```

|     |         |      |     |     |      |     |     |
|-----|---------|------|-----|-----|------|-----|-----|
| 219 | 01JAN98 | 9:31 | LGA | LON | 3442 | 198 | 250 |
|-----|---------|------|-----|-----|------|-----|-----|

<...10,000 more observations...>

```
proc sql;
create table ora_air.flights98
(BULKLOAD=YES BL_OPTIONS='ERRORS=899,LOAD=5000') as
  select * from sasflt.flt98;
quit;
```

During a load, certain SQL\*Loader files are created, such as the data, log, and control files. Unless otherwise specified, they are given a default name and written to the temporary file directory that the UTILLOC= system option specifies. (For details about this option, see [SAS System Options: Reference](#).) For this example, the default names are bl\_flights98.dat, bl\_flights98.log, and bl\_flights98.ctl.

## Locking in the Oracle Interface

These LIBNAME and data set options let you control how the Oracle interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134.

**READ\_LOCK\_TYPE= NOLOCK | ROW | TABLE**  
Default: NOLOCK

Here are the valid values for this option:

- NOLOCK — table locking is not used during the reading of tables and views.
- ROW — the Oracle ROW SHARE table lock is used during the reading of tables and views.
- TABLE — the Oracle SHARE table lock is used during the reading of tables and views.

If you set READ\_LOCK\_TYPE= to either TABLE or ROW, you must also set the CONNECTION= option to UNIQUE. If not, an error occurs.

**UPDATE\_LOCK\_TYPE= NOLOCK | ROW | TABLE**  
Default: NOLOCK

Here are the valid values for this option:

- ROW — the Oracle ROW SHARE table lock is used during the reading of tables and views for update.
- TABLE — the Oracle EXCLUSIVE table lock is used during the reading of tables and views for update.
- NOLOCK — table locking is not used during the reading of tables and views for update.
  - If OR\_UPD\_NOWHERE=YES, updates are performed using serializable transactions.
  - If OR\_UPD\_NOWHERE=NO, updates are performed using an extra WHERE clause to ensure that the row has not been updated since it was

first read. Updates might fail under these conditions, because other users might modify a row after the row was read for update.

#### `READ_ISOLATION_LEVEL= READCOMMITTED | SERIALIZABLE`

Oracle supports the READCOMMITTED and SERIALIZABLE read isolation levels, as specified in the table below. The SPOOL= option overrides the `READ_ISOLATION_LEVEL`= option. The `READ_ISOLATION_LEVEL`= option should be rarely needed because the SAS/ACCESS engine chooses the appropriate isolation level based on other locking options.

*Table 32.4 Isolation Levels for Oracle*

| Isolation Level | Definition                                                                   |
|-----------------|------------------------------------------------------------------------------|
| SERIALIZABLE    | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.           |
| READCOMMITTED   | Does not allow dirty Reads; does allow nonrepeatable Reads and phantom Reads |

#### `UPDATE_ISOLATION_LEVEL= READCOMMITTED | SERIALIZABLE`

Oracle supports the READCOMMITTED and SERIALIZABLE isolation levels, as specified in the preceding table, for updates.

Default: READCOMMITTED

This option should be rarely needed because the SAS/ACCESS engine chooses the appropriate isolation level based on other locking options.

## Naming Conventions for Oracle

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

The `PRESERVE_TAB_NAMES=` and `PRESERVE_COL_NAMES=` options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Oracle is case sensitive, and all names default to uppercase.

You can name such Oracle objects as tables, views, columns, and indexes. They follow these naming conventions.

- A name must be from 1 to 30 characters long. Database names are limited to 8 characters, and link names are limited to 128 characters.
- A name must begin with a letter. However, if you enclose the name in double quotation marks, it can begin with any character.
- A name can contain the letters A through Z, the digits 0 through 9, the underscore (\_), \$, and #. If the name appears within double quotation marks, it can contain any characters, except double quotation marks.

- To preserve special characters in the user name (`USER=`) in the LIBNAME statement, you must enclose the name in single quotation marks and then enclose everything in double quotation marks.
- Names are not case sensitive. For example, `CUSTOMER` and `Customer` are the same. However, if you enclose an object name in double quotation marks, it is case sensitive.
- A name cannot be an Oracle reserved word.
- A name cannot be the same as another Oracle object in the same schema.

## Data Types for Oracle

### Overview

Every column in a table has a name and a data type. The data type tells Oracle how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Oracle data types, nulls, default values, and data conversions.

## Supported Oracle Data Types

Here are the data types that the Oracle engine supports.

- Character data:

|                                            |                          |
|--------------------------------------------|--------------------------|
| <code>CHAR (n)</code>                      | <code>NVARCHAR(n)</code> |
| <code>CLOB (character large object)</code> | <code>VARCHAR2(n)</code> |
| <code>NCHAR(n)</code>                      |                          |

- Numeric data:

|                            |                          |
|----------------------------|--------------------------|
| <code>BINARY_DOUBLE</code> | <code>NUMBER(p)</code>   |
| <code>BINARY_FLOAT</code>  | <code>NUMBER(p,s)</code> |
| <code>NUMBER</code>        |                          |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Date, timestamp, and interval data:

|                                |                        |
|--------------------------------|------------------------|
| DATE, TIMESTAMP                | INTERVAL YEAR TO MONTH |
| TIMESTAMP WITH TIME ZONE       | INTERVAL DAY TO SECOND |
| TIMESTAMP WITH LOCAL TIME ZONE |                        |

- Binary data:

RAW(*n*)    BLOB

SAS/ACCESS Interface to Oracle does not support the Oracle MLSLABEL data type.

For compatibility with other DBMSs, Oracle supports the syntax for a wide variety of numeric data types, including DECIMAL, INTEGER, REAL, DOUBLE-PRECISION, and SMALLINT. All forms of numeric data types are actually stored in the same internal Oracle NUMBER format. The additional numeric data types are variations of precision and scale. A null scale implies a floating-point number, and a non-null scale implies a fixed-point number.

For detailed information about Oracle data types, see your Oracle documentation.

## Example 1: Timestamp

```
%let PTCOMP= %str(user=myusr1 pw=mypwd1 path=mysrv1);
%let MYCONN= %str(user=myusr1 pw=mypwd1 path=mysrv1);

options sastrace=",,," sastraceloc=saslog nostsuffix;

proc sql;
connect to oracle ( &PTCONN);

/* Execute ( drop table EMP_ATTENDANCE) by oracle;*/
execute ( create table EMP_ATTENDANCE ( EMP_NAME VARCHAR2(10),
arrival_timestamp TIMESTAMP, departure_timestamp TIMESTAMP ) ) by
oracle;
execute ( insert into EMP_ATTENDANCE values
('John Doe', systimestamp, systimestamp+.2) ) by oracle;
execute ( insert into EMP_ATTENDANCE values
('Sue Day', TIMESTAMP'1980-1-12 10:13:23.33',
TIMESTAMP'1980-1-12 17:13:23.33') ) by oracle;
quit;

libname ora oracle &MYCONN;

proc contents data=ora.EMP_ATTENDANCE; run;

proc sql;
/* Read TIMESTAMP data type */
select * from ora.EMP_ATTENDANCE;
quit;

/* Append to TIMESTAMP data type */
data work.new;
EMP_NAME='New Bee1';
```

```

ARRIVAL_TIMESTAMP='30sep1998:14:00:35.00'dt;
DEPARTURE_TIMESTAMP='30sep1998:17:00:14.44'dt;  output;
EMP_NAME='New Bee2';
ARRIVAL_TIMESTAMP='30sep1998:11:00:25.11'dt;
DEPARTURE_TIMESTAMP='30sep1998:14:00:35.27'dt;  output;
EMP_NAME='New Bee3';
ARRIVAL_TIMESTAMP='30sep1998:08:00:35.33'dt;
DEPARTURE_TIMESTAMP='30sep1998:17:00:35.10'dt;  output;
format ARRIVAL_TIMESTAMP datetime23.2;
format DEPARTURE_TIMESTAMP datetime23.2;
run;

title2 'After append';
proc append data=work.new base=ora.EMP_ATTENDANCE ; run;
proc print data=ora.EMP_ATTENDANCE ; run;

/* Update TIMESTAMP data type */
proc sql;
update ora.EMP_ATTENDANCE set ARRIVAL_TIMESTAMP=.
  where EMP_NAME like '%Bee2%' ;

select * from ora.EMP_ATTENDANCE ;

delete from ora.EMP_ATTENDANCE where EMP_NAME like '%Bee2%' ;

select * from ora.EMP_ATTENDANCE ;

/* OUTPUT: Creating a brand new table using Data Step*/
data work.sasdsfsec; c_ts='30sep1998:14:00:35.16'dt; k=1; output;
  c_ts='.'dt; k=2; output;
  format c_ts datetime23.2; run;

/* picks default TIMESTAMP type */
options sastrace=",,d" sastraceloc=saslog nostsuffix;
data ora.tab_tsfsec; set work.sasdsfsec; run;
options sastrace=",,," sastraceloc=saslog nostsuffix;
proc datasets library=ora;
  delete tab_tsfsec;run;

/* Override the default data type */
options sastrace=",,d" sastraceloc=saslog nostsuffix;
data ora.tab_tsfsec (dbtype=(c_ts='timestamp(3)'));
  c_ts='30sep1998:14:00:35'dt;
  format c_ts datetime23.; run;
options sastrace=",,," sastraceloc=saslog nostsuffix;
proc datasets library=ora;
  delete tab_tsfsec;run;

proc print data=ora.tab_tsfsec; run;

/* Output: Create a new table with bulkload=yes */
title2 'Test OUTPUT with bulkloader';
proc datasets library=ora;
  delete tab_tsfsec;run;

/* Select default TIMESTAMP type */

```

```
data ora.tab_tsfsec (bulkload=yes); set work.sasdsfsec; run;
proc print data=ora.tab_tsfsec;run;
```

## Example 2: Interval Year to Month

```
proc sql;
connect to oracle ( &PTCONN);
execute ( drop table PRODUCT_INFO) by oracle;

execute (
    create table PRODUCT_INFO ( PRODUCT VARCHAR2(20),
        LIST_PRICE number(8,2),
        WARRANTY_PERIOD INTERVAL YEAR(2) TO MONTH ))by oracle;
execute (
    insert into PRODUCT_INFO values ('Dish Washer', 4000, '02-00')
)by Oracle;
execute (
    insert into PRODUCT_INFO values ('TV', 6000, '03-06'))by Oracle;
quit;

proc contents data=ora.PRODUCT_INFO;run;
/* Show WARRANTY_PERIOD as number of months */
proc print data=ora.PRODUCT_INFO; run;

/* Show WARRANTY_PERIOD in a format as in Oracle*/
proc print
    data=ora.PRODUCT_INFO(dbsastype=(WARRANTY_PERIOD='CHAR (6)' )); run;

/* Add a new product */
data new_prods;
    PRODUCT='Dryer';  LIST_PRICE=2000;WARRANTY_PERIOD=12;
run;

proc sql;
insert into ora.PRODUCT_INFO select * from new_prods;
select * from ora.PRODUCT_INFO;
select * from ora.PRODUCT_INFO where WARRANTY_PERIOD > 24;
quit;
```

## Example 3: Interval Day to Second

```
proc sql;
connect to oracle ( &PTCONN);
execute ( drop table PERF_TESTS) by oracle;

execute (
    create table PERF_TESTS ( TEST_NUMBER number(4) primary key,
        TIME_TAKEN INTERVAL DAY TO SECOND ))by oracle;

execute (
```

```

insert into PERF_TESTS
  values (1, '0 00:01:05.000200000'))by Oracle;

execute (
  insert into PERF_TESTS values (2, '0 00:01:03.400000000'))by Oracle;
quit;

proc contents data=ora.PERF_TESTS; run;

/* Show TIME_TAKEN as number of seconds */
proc print data=ora.PERF_TESTS; run;

/* Show TIME_TAKEN in a format just like in Oracle*/
proc print
  data=ora.PERF_TESTS(dbsastype=(TIME_TAKEN='CHAR(25)')); run;

/* Add a new test*/
data new_tests;
  TEST_NUMBER=3; TIME_TAKEN=50;
run;

proc sql;
insert into ora.PERF_TESTS select * from new_tests;
select * from ora.PERF_TESTS;

select * from ora.PERF_TESTS where TIME_TAKEN < 60;
quit;

```

## Default Data Types

### Oracle Null and Default Values

Oracle has a special value called NULL. An Oracle NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads an Oracle NULL value, it interprets it as a SAS missing value.

By default, Oracle columns accept NULL values. However, you can specify columns so that they cannot contain NULL data. NOT NULL tells Oracle not to add a row to the table unless the row has a value for that column. When creating an Oracle table with SAS/ACCESS, you can use the [DBNULL= data set option](#) to indicate whether NULL is a valid value for specified columns.

To control how SAS missing character values are handled, use the [NULLCHAR=](#) and [NULLCHARVAL=](#) data set options.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

## Default Data Types for SAS Output

The table below shows the default data types for SAS character variables based on the length of the variable and the Oracle version.

| Oracle Server | Character Variable Length   | Data Type |
|---------------|-----------------------------|-----------|
| Prior to 12c  | less than or equal to 4000  | VARCHAR2  |
|               | greater than 4000           | CLOB      |
| 12c and later | less than or equal to 32767 | VARCHAR2  |
|               | greater than 32767          | CLOB      |

The table below shows the default data types for SAS character variables when the NOTRANSCODE attribute is specified.

| Oracle Server | Character Variable Length   | Data Type |
|---------------|-----------------------------|-----------|
| Prior to 12c  | less than or equal to 2000  | RAW       |
|               | greater than 2000           | BLOB      |
| 12c and later | less than or equal to 32767 | RAW       |
|               | greater than 32767          | BLOB      |

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Oracle assigns to SAS variables when using the [LIBNAME statement](#) to read from an Oracle table. These default formats are based on Oracle column attributes.

*Table 32.5 LIBNAME Statement: Default SAS Formats for Oracle Data Types*

| Oracle Data Type                  | Default SAS Format |
|-----------------------------------|--------------------|
| CHAR( <i>n</i> ) <sup>1</sup>     | \$ <i>w</i> .      |
| NCHAR( <i>n</i> ) <sup>1</sup>    | \$ <i>w</i> .      |
| NVARCHAR( <i>n</i> ) <sup>1</sup> | \$ <i>w</i> .      |

| Oracle Data Type              | Default SAS Format                                                                                             |
|-------------------------------|----------------------------------------------------------------------------------------------------------------|
| VARCHAR2( <i>n</i> )          | \$ <i>w</i> .                                                                                                  |
| LONG                          | \$ <i>w</i> . (where <i>w</i> is the minimum of 32767 and the value of the DBMAX_TEXT= option)                 |
| CLOB                          | \$ <i>w</i> . <sup>1</sup><br>(where <i>w</i> is the minimum of 32767 and the value of the DBMAX_TEXT= option) |
| RAW( <i>n</i> )               | \$HEX <i>w</i> . <sup>1</sup><br>(where <i>w</i> is $2^*n$ )                                                   |
| LONG RAW                      | \$HEX <i>w</i> . (where <i>w</i> /2 is the minimum of 32767 and the value of the DBMAX_TEXT= option)           |
| BLOB RAW                      | \$HEX <i>w</i> . (where <i>w</i> /2 is the minimum of 32767 and the value of the DBMAX_TEXT= option)           |
| BINARY_DOUBLE                 | none                                                                                                           |
| BINARY_FLOAT                  | none                                                                                                           |
| NUMBER                        | none                                                                                                           |
| NUMBER( <i>p</i> )            | <i>w</i> .                                                                                                     |
| NUMBER( <i>p,s</i> )          | <i>w.d</i>                                                                                                     |
| DATE                          | DATETIME20.                                                                                                    |
| TIMESTAMP                     | DATETIME <i>w.d</i> (where <i>d</i> is derived from the fractional-second precision)                           |
| TIMESTAMP WITH LOCAL TIMEZONE | DATETIME <i>w.d</i> (where <i>d</i> is derived from the fractional-second precision)                           |
| TIMESTAMP WITH TIMEZONE       | \$ <i>w</i> .                                                                                                  |
| INTERVAL YEAR TO MONTH        | <i>w</i> . (where <i>w</i> is derived from the year precision)                                                 |
| INTERVAL DAY TO SECOND        | <i>w.d</i> (where <i>w</i> is derived from the fractional-second precision)                                    |

<sup>1</sup> The value of the “DBMAX\_TEXT= LIBNAME Statement Option” option can override these values.

SAS/ACCESS does not support Oracle data types that do not appear in this table.

If Oracle data falls outside valid SAS data ranges, the values are usually counted as missing.

SAS automatically converts Oracle NUMBER types to SAS number formats by using an algorithm that determines the correct scale and precision. When the scale and precision cannot be determined, SAS/ACCESS allows the procedure or application to determine the format. You can also convert numeric data to character data by using the SQL pass-through facility with the Oracle TO\_CHAR function. See your Oracle documentation for more details.

The table below shows the default Oracle data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

*Table 32.6 LIBNAME Statement: Default Oracle Data Types for SAS Formats*

| SAS Variable Format                                                     | Oracle Data Type |
|-------------------------------------------------------------------------|------------------|
| \$w.                                                                    | VARCHAR2(w)      |
| \$w. (where w > 4000)                                                   | CLOB             |
| w.d                                                                     | NUMBER(p,s)      |
| any date, time, or datetime format without fractional parts of a second | DATE             |
| any date, time, or datetime format without fractional parts of a second | TIMESTAMP        |

To override these data types, use the DBTYPE= data set option during output processing.

## ACCESS Procedure Data Conversions

This table shows the default SAS variable formats that SAS/ACCESS assigns to Oracle data types when you use the [ACCESS procedure](#).

*Table 32.7 PROC ACCESS: Default SAS Formats for Oracle Data Types*

| Oracle Data Type | Default SAS Format               |
|------------------|----------------------------------|
| CHAR(n)          | \$n. (n <= 200) \$200. (n > 200) |
| VARCHAR2(n)      | \$n. (n <= 200) \$200. (n > 200) |
| FLOAT            | BEST22.                          |
| NUMBER           | BEST22.                          |

| Oracle Data Type | Default SAS Format                      |
|------------------|-----------------------------------------|
| NUMBER( $p$ )    | w.                                      |
| NUMBER( $p, s$ ) | w.d                                     |
| DATE             | DATETIME16.                             |
| CLOB             | \$200.                                  |
| RAW( $n$ )       | \$n. ( $n < 200$ ) \$200. ( $n > 200$ ) |
| BLOB RAW         | \$200.                                  |

Oracle data types that are omitted from this table are not supported by SAS/ACCESS. If Oracle data falls outside valid SAS data ranges, the values are usually counted as missing.

This table shows the correlation between the Oracle NUMBER data types and the default SAS formats that are created from that data type.

*Table 32.8 Default SAS Formats for Oracle NUMBER Data Types*

| Oracle NUMBER Data Type | Rules                      | Default SAS Format         |
|-------------------------|----------------------------|----------------------------|
| NUMBER( $p$ )           | $0 < p \leq 32$            | ( $p + 1$ ).0              |
| NUMBER( $p, s$ )        | $p > 0, s < 0,  s  < p$    | ( $p +  s  + 1$ ).0        |
| NUMBER( $p, s$ )        | $p > 0, s < 0,  s  \geq p$ | ( $p +  s  + 1$ ).0        |
| NUMBER( $p, s$ )        | $p > 0, s > 0, s < p$      | ( $p + 2$ ).s              |
| NUMBER( $p, s$ )        | $p > 0, s > 0, s \geq p$   | ( $s + 3$ ).s              |
| NUMBER( $p$ )           | $p > 32$                   | BEST22. SAS selects format |
| NUMBER                  | $p, s$ unspecified         | BEST22. SAS selects format |

The general form of an Oracle number is NUMBER( $p, s$ ) where  $p$  is the precision and  $s$  is the scale of the number. Oracle specifies precision as the total number of digits, with a valid range of -84 to 127. However, a negative scale means that the number is rounded to the specified number of places to the left of the decimal. For example, if the number 1,234.56 is specified as data type NUMBER(8,-2), it is rounded to the nearest hundred and stored as 1,200.

---

## DBLOAD Procedure Data Conversions

This table shows the default Oracle data types that SAS/ACCESS assigns to SAS variable formats when you use the [DBLOAD procedure](#).

*Table 32.9 PROC DBLOAD: Default Oracle Data Types for SAS Formats*

| SAS Variable Format             | Oracle Data Type     |
|---------------------------------|----------------------|
| \$w.                            | CHAR( <i>n</i> )     |
| w.                              | NUMBER( <i>p</i> )   |
| w. <i>d</i>                     | NUMBER( <i>p,s</i> ) |
| all other numerics <sup>1</sup> | NUMBER               |
| datetimew. <i>d</i>             | DATE                 |
| datew.                          | DATE                 |
| time. <sup>2</sup>              | NUMBER               |

<sup>1</sup> Includes all SAS numeric formats, such as BINARY8 and E10.0.

<sup>2</sup> Includes all SAS time formats, such as TODw,*d* and HHMMw,*d*.

---

## Sample Programs for Oracle

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to PostgreSQL

---

|                                                                   |      |
|-------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to PostgreSQL</i> | 1172 |
| <i>Introduction to SAS/ACCESS Interface to PostgreSQL</i>         | 1172 |
| <i>LIBNAME Statement for the PostgreSQL Engine</i>                | 1173 |
| Overview                                                          | 1173 |
| Arguments                                                         | 1173 |
| LIBNAME Statement Examples for PostgreSQL                         | 1177 |
| <i>Data Set Options for PostgreSQL</i>                            | 1178 |
| <i>Best Practices When You Connect to CockroachDB</i>             | 1180 |
| <i>SQL Pass-Through Facility Specifics for PostgreSQL</i>         | 1181 |
| Key Information                                                   | 1181 |
| CONNECT Statement Examples for PostgreSQL                         | 1181 |
| <i>Temporary Table Support for PostgreSQL</i>                     | 1182 |
| <i>Running PROC COPY In-Database for PostgreSQL</i>               | 1182 |
| Details about Running PROC COPY for PostgreSQL                    | 1182 |
| PROC COPY Examples                                                | 1183 |
| <i>Passing SAS Functions to PostgreSQL</i>                        | 1184 |
| <i>Passing Joins to PostgreSQL</i>                                | 1185 |
| <i>Bulk Loading and Unloading for PostgreSQL</i>                  | 1186 |
| Overview of Bulk Loading and Unloading for PostgreSQL             | 1186 |
| Data Set Options with Bulk Loading and Unloading                  | 1186 |
| Using the PSQL Tool for Bulk Loading and Unloading                | 1187 |
| Bulk-Load Example for PostgreSQL                                  | 1187 |
| <i>Improve Performance When Deleting or Updating Rows</i>         | 1187 |
| <i>Locking in the PostgreSQL Interface</i>                        | 1188 |
| <i>Working with NLS Characters in PostgreSQL Data</i>             | 1189 |
| <i>Naming Conventions for PostgreSQL</i>                          | 1189 |
| <i>Data Types for PostgreSQL</i>                                  | 1190 |

|                                                        |      |
|--------------------------------------------------------|------|
| Overview of Data Types .....                           | 1190 |
| PostgreSQL Null Values .....                           | 1190 |
| Working with Long Character Values in PostgreSQL ..... | 1191 |
| LIBNAME Statement Data Conversions .....               | 1192 |
| <i>Sample Programs for PostgreSQL</i> .....            | 1194 |

---

# System Requirements for SAS/ACCESS Interface to PostgreSQL

You can find information about system requirements for SAS/ACCESS Interface to PostgreSQL in the following locations:

- [System Requirements for SAS/ACCESS Interface to PostgreSQL with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

# Introduction to SAS/ACCESS Interface to PostgreSQL

For available SAS/ACCESS features, see “[SAS/ACCESS Interface to PostgreSQL: Supported Features](#)” on page 113. For more information about PostgreSQL, see your PostgreSQL documentation.

**Note:** The SAS/ACCESS Interface to PostgreSQL was implemented for [SAS 9.4](#).

Beginning in [SAS Viya 3.3](#), SAS/ACCESS Interface to PostgreSQL includes SAS Data Connector to PostgreSQL. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“PostgreSQL Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

# LIBNAME Statement for the PostgreSQL Engine

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to PostgreSQL supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing PostgreSQL.

**LIBNAME** *libref* **postgres** <*connection-options*> <*LIBNAME-options*>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *postgres*

specifies the SAS/ACCESS engine name for the PostgreSQL interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are specified.

---

**Note:** All of these connection options (preceding the table of PostgreSQL LIBNAME statement options) are valid when used in the CONNECT statement with the SQL procedure.

---

### *SERVER=<'>PostgreSQL-server-name<'>*

specifies the server name or IP address of the PostgreSQL server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

### *DATABASE=<'>PostgreSQL-database-name<'>*

specifies the name of the database on the PostgreSQL server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORT=port**

specifies the port number that is used to connect to the specified PostgreSQL server.

Default: 5432

**USER=<'>PostgreSQL-user-name<'>**

specifies the PostgreSQL user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: UID=

**PASSWORD=<'>PostgreSQL-password<'>**

specifies the password that is associated with your PostgreSQL user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: PASS=, PW=, PWD=

**DSN=<'>PostgreSQL-data-source<'>**

specifies the configured PostgreSQL ODBC data source to which you want to connect. Use this option if you have existing PostgreSQL ODBC data sources that are configured on your client. This method requires additional setup—either through the Administrator control panel on Windows platforms or through the odbc.ini file or a similarly named configuration file on UNIX platforms. So it is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

**COMPLETE=<'>CLI-connection-string<'>**

specifies connection information for your database for PCs only. Separate multiple options with a semicolon. When a successful connection is made, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, you are prompted with a dialog box that displays the values from the COMPLETE= connection string. You can edit any field before you connect to the database. See your PostgreSQL documentation for more details.

This option is not available on UNIX platforms.

Support for this connection option was added in SAS Viya 3.4.

**PROMPT=<'>PostgreSQL-connection-options<'>**

specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately try to connect to the DBMS. Instead, it displays a dialog box in SAS Display Manager that contains the values that you entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the database.

Restriction: This option is not available on UNIX platforms.

Support for this connection option was added in SAS Viya 3.4.

**NOPROMPT=<'>PostgreSQL-connection-options<'>**

specifies connection options for your database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned. No dialog box is displayed to help you with the connection string.

Support for this connection option was added in SAS Viya 3.4.

*LIBNAME-options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to PostgreSQL with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

*Table 33.1 SAS/ACCESS LIBNAME Options for PostgreSQL*

| Option               | Default Value                                                                        | Valid in CONNECT |
|----------------------|--------------------------------------------------------------------------------------|------------------|
| ACCESS=              | none                                                                                 |                  |
| AUTHDOMAIN=          | none                                                                                 | •                |
| AUTOCOMMIT=          | NO                                                                                   | •                |
| CLIENT_ENCODING=     | current SAS session encoding                                                         |                  |
| CONNECTION=          | SHAREDREAD                                                                           | •                |
| CONNECTION_GROUP=    | none                                                                                 | •                |
| CONOPTS=             | none                                                                                 | •                |
| CURSOR_TYPE=         | DYNAMIC                                                                              | •                |
| DBCLIENT_MAX_BYTES=  | matches the maximum number of bytes per single character of the SAS session encoding | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows)                                   |                  |
| DBCONINIT=           | none                                                                                 | •                |
| DBCONTERM=           | none                                                                                 | •                |
| DBCREATE_TABLE_OPTS= | none                                                                                 |                  |
| DBGEN_NAME=          | DBMS                                                                                 | •                |
| DBINDEX=             | NO                                                                                   |                  |
| DBLIBINIT=           | none                                                                                 |                  |

| Option                    | Default Value                                               | Valid in CONNECT |
|---------------------------|-------------------------------------------------------------|------------------|
| DBLIBTERM=                | none                                                        |                  |
| DBMAX_TEXT=               | 1024                                                        | •                |
| DBMSTEMP=                 | NO                                                          |                  |
| DBNULLKEYS=               | YES                                                         |                  |
| DBPROMPT=                 | NO                                                          | •                |
| DEFER=                    | NO                                                          | •                |
| DELETE_MULT_ROWS=         | NO                                                          | •                |
| DIRECT_EXE=               | none                                                        |                  |
| DIRECT_SQL=               | YES                                                         |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                          |                  |
| INSERT_SQL=               | YES                                                         |                  |
| INSERTBUFF=               | 1                                                           |                  |
| KEYSET_SIZE=              | 0                                                           | •                |
| LOGIN_TIMEOUT=            | 0                                                           | •                |
| MULTI_DATASRC_OPT=        | NONE                                                        |                  |
| POST_STMT_OPTS=           | none                                                        |                  |
| PRESERVE_COL_NAMES=       | NO                                                          |                  |
| PRESERVE_TAB_NAMES=       | NO                                                          |                  |
| QUALIFIER=                | none                                                        |                  |
| QUERY_TIMEOUT=            | 0                                                           | •                |
| QUOTE_CHAR=               | none                                                        | •                |
| READ_ISOLATION_LEVEL=     | RC (see “Locking in the PostgreSQL Interface” on page 1188) | •                |
| READ_LOCK_TYPE=           | ROW                                                         | •                |

| Option                  | Default Value                                               | Valid in CONNECT |
|-------------------------|-------------------------------------------------------------|------------------|
| READBUFF=               | 0                                                           | •                |
| REREAD_EXPOSURE=        | NO                                                          | •                |
| SCHEMA=                 | none                                                        |                  |
| SPOOL=                  | YES                                                         |                  |
| SQL_FUNCTIONS=          | none                                                        |                  |
| SQL_FUNCTIONS_COPY=     | none                                                        |                  |
| SQLGENERATION=          | none                                                        |                  |
| SSLMODE=                | prefer                                                      |                  |
| STRINGDATES=            | NO                                                          | •                |
| SUB_CHAR=               | none                                                        |                  |
| TRACE=                  | NO                                                          | •                |
| TRACEFILE=              | none                                                        | •                |
| UPDATE_ISOLATION_LEVEL= | RC (see “Locking in the PostgreSQL Interface” on page 1188) | •                |
| UPDATE_LOCK_TYPE=       | ROW                                                         | •                |
| UTILCONN_TRANSIENT=     | NO                                                          |                  |

## LIBNAME Statement Examples for PostgreSQL

In this example, SERVER=, DATABASE=, USER=, and PASSWORD= are connection options. No DSN style is specified. This is the default method, which is recommended.

```
libname A1 postgres server=mysrv1 port=5432
      user=myusr1 password='mypwd1' database=mydb1;
```

This example requires that you specify a DSN style.

```
libname B1 postgres dsn=ptgtest user=myusr1 password=mypwd1;
```

# Data Set Options for PostgreSQL

All SAS/ACCESS data set options in this table are supported for PostgreSQL. Default values are provided where applicable. For more information, see [Data Set Options for Relational Databases on page 370](#).

**Table 33.2** SAS/ACCESS Data Set Options for PostgreSQL

| Option               | Default Value                                        |
|----------------------|------------------------------------------------------|
| BL_DATAFILE=         | none                                                 |
| BL_DEFAULT_DIR=      | <i>temporary-file-directory</i>                      |
| BL_DELETE_DATAFILE=  | YES                                                  |
| BL_DELIMITER=        | (pipe)                                               |
| BL_ESCAPE=           | \                                                    |
| BL_FORMAT=           | TEXT                                                 |
| BL_LOAD_METHOD=      | APPEND (table with data)                             |
| BL_LOGFILE=          | none                                                 |
| BL_NULL=             | '\N' [TEXT mode], unquoted empty value<br>[CSV mode] |
| BL_PSQL_PATH=        | psql                                                 |
| BL_QUOTE=            | " (double quotation mark)                            |
| BULKLOAD=            | NO                                                   |
| BULKUNLOAD=          | NO                                                   |
| CURSOR_TYPE=         | LIBNAME option value                                 |
| DBCOMMIT=            | LIBNAME option value                                 |
| DBCONDITION=         | none                                                 |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                                 |
| DBFORCE=             | NO                                                   |

| <b>Option</b>             | <b>Default Value</b>                         |
|---------------------------|----------------------------------------------|
| DBGEN_NAME=               | DBMS                                         |
| DBINDEX=                  | LIBNAME option value                         |
| DBKEY=                    | none                                         |
| DBLABEL=                  | NO                                           |
| DBLARGETABLE=             | none                                         |
| DBMAX_TEXT=               | 1024                                         |
| DBNULL=                   | YES                                          |
| DBNULLKEYS=               | LIBNAME option value                         |
| DBPROMPT=                 | LIBNAME option value                         |
| DBSASLABEL=               | COMPAT                                       |
| DBSASTYPE=                | see “Data Types for PostgreSQL” on page 1190 |
| DBTYPE=                   | see “Data Types for PostgreSQL” on page 1190 |
| ERRLIMIT=                 | 1                                            |
| IGNORE_READ_ONLY_COLUMNS= | NO                                           |
| INSERT_SQL=               | none                                         |
| INSERTBUFF=               | LIBNAME option value                         |
| KEYSET_SIZE=              | LIBNAME option value                         |
| NULLCHAR=                 | SAS                                          |
| NULLCHARVAL=              | a blank character                            |
| PG_IDENTITY_COLS=         | none                                         |
| POST_STMT_OPTS=           | none                                         |
| POST_TABLE_OPTS=          | none                                         |
| PRE_STMT_OPTS=            | none                                         |

| Option                  | Default Value        |
|-------------------------|----------------------|
| PRE_TABLE_OPTS=         | none                 |
| PRESERVE_COL_NAMES=     | LIBNAME option value |
| QUALIFIER=              | LIBNAME option value |
| QUERY_TIMEOUT=          | LIBNAME option value |
| READBUFF=               | LIBNAME option value |
| READ_LOCK_TYPE=         | ROW                  |
| SASDATEFMT=             | none                 |
| SCHEMA=                 | LIBNAME option value |
| SUB_CHAR=               | none                 |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |
| UPDATE_LOCK_TYPE=       | LIBNAME option value |

---

## Best Practices When You Connect to CockroachDB

CockroachDB is a cloud-based data source that is compatible with a PostgreSQL LIBNAME connection. When you connect to CockroachDB, be aware of the following differences in behavior from a standard connection to PostgreSQL:

- The best practice when you connect to CockroachDB is to use the default SHAREDREAD value for the CONNECTION= LIBNAME option. Setting CONNECTION=SHARED is not fully supported, although you can use this value for Read operations. CREATE or DROP operations are not supported when you set CONNECTION=SHARED.
- It is recommended that you limit table names and column names to 128 bytes. CockroachDB documentation states that this name limit might be enforced in future updates. Also, accessing table or columns names longer than 128 bytes is not currently supported for PROC FEDSQL.
- When you use NLS data on CockroachDB, the best practice is to use the UTF-8 SAS session encoding.

---

# SQL Pass-Through Facility Specifics for PostgreSQL

---

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the PostgreSQL interface.

- The *dbms-name* is **POSTGRES**.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to PostgreSQL. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default POSTGRES alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).

---

## CONNECT Statement Examples for PostgreSQL

---

This example connects to PostgreSQL and then disconnects from it.

```
proc sql noerrorstop;
    connect to postgres as x1(server=mysrv1 port=5432
                           user=mysurl password='mypwd1' database=mydb1);
    disconnect from x1;
    quit;
```

This next example connects to PostgreSQL, executes some SQL statements, and then disconnects from PostgreSQL.

```
proc sql noerrorstop;
    connect to postgres as x1(server=mysrv1 port=5432
                           user=mysurl password='mypwd1' database=mydb1);

    execute ( CREATE TABLE t1 ( no int primary key, state varchar(10) ) )
    by x1;
    execute ( INSERT INTO t1 values (1, 'USA') ) by x1;
    execute ( INSERT INTO t1 values (2, 'CHN') ) by x1;
    select * from connection to x1 (SELECT * FROM t1 ORDER BY no);

    disconnect from x1;
    quit;
```

# Temporary Table Support for PostgreSQL

SAS/ACCESS Interface to PostgreSQL supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## Running PROC COPY In-Database for PostgreSQL

### Details about Running PROC COPY for PostgreSQL

Copying data in-database results in faster copy times because the data does not have to be loaded into or moved back out of SAS. You can run the COPY procedure in-database to copy PostgreSQL data in these situations:

- to copy data from one schema to another. In this case, the librefs for the IN= and OUT= options for PROC COPY must match for the SERVER=, PORT=, and DATABASE= values.
- to copy data from one database to another. This case also requires that you have enabled the `postgres_fdw` extension for the OUT= location. For more information, see your PostgreSQL documentation.

In all cases, to copy data in-database, the USER= who you specify for the source location must have Read permission for that location. Also, the USER= for the target location must have Read permission for the source location and Write permission for the target location.

**Note:** Copying data in-database is also supported for the COPY operation in PROC DATASETS.

Here are some points to keep in mind when you copy data in-database:

- You can specify the ACCEL | NOACCEL option in the PROC COPY statement when you copy data. The ACCEL option specifies that data is copied in-database. By default, ACCEL is enabled.
- You can also specify the REPLACE | NOREREPLACE system option in the OPTIONS statement when you use the COPY procedure. The REPLACE | NOREREPLACE system option specifies whether to replace a data set if it already exists. The REPLACE option is enabled by default.

**Note:** When the REPLACE system option is enabled, an existing target table is deleted before PROC COPY attempts to copy data.

- In-database copy is not supported if the input library is a concatenated library. You can copy to a concatenated library if the first target library matches the input library.
- The OVERRIDE= and CLONE | NOCLONE options in the COPY statement are ignored when copying data in-database.
- You can specify PRESERVE\_NAMES=YES in the LIBNAME statement to preserve the case of table or column names. This also enables you to copy names with special characters. PRESERVE\_NAMES= is an alias for both the PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= LIBNAME options. This option is recommended because PostgreSQL is a case-sensitive DBMS.
- Data types that are not supported in SAS Viya, such as BINARY or DECIMAL, are retained when you copy data in-database. If the data is moved into and out of SAS Viya (without copying data in-database), then data types or attributes are converted to types and attributes that are supported in SAS Viya.
- Indexes and integrity constraints are copied with a table when it is copied in-database. These are not copied if the table is moved into and out of SAS Viya (without copying data in-database) during the copy operation.

To verify whether the COPY procedure ran in-database, specify MSGLEVEL=i in the OPTIONS statement.

---

## See Also

- [“COPY Procedure” in \*Base SAS Procedures Guide\*](#)
- [“DATASETS Procedure” in \*Base SAS Procedures Guide\*](#)
- [“MSGLEVEL= System Option” in \*SAS System Options: Reference\*](#)
- [“REPLACE System Option” in \*SAS System Options: Reference\*](#)

---

## PROC COPY Examples

This example shows how to copy data in PostgreSQL between schemas in a database. This example copies the table my\_table from schema inschema into the schema outsch. The case of the table name is preserved because you specify PRESERVE\_NAMES=YES. By specifying the REPLACE system option, you ensure that if the destination table already exists, it is replaced by the copy procedure. The ACCEL option for the PROC COPY statement is enabled by default. This option controls whether data is copied in-database.

```
libname pgin postgres user="myuser" database=userdb port=5432
      server="pgserver.com" password=mypwd schema=inschema
      preserve_names=yes;
libname pgout postgres user="myuser" database=userdb port=5432
      server="pgserver.com" password=mypwd schema=outsch
```

```

preserve_names=yes;

options replace;

proc copy in=pgin out=pgout;
  select my_table;
run;
quit;

proc contents data=pgout.my_table; run;

```

In the following example, you see how to copy table project\_tab between different databases. The databases reside on different servers. For the copy to work, the user myuser must have permission to create a schema in the Pgout libref. The ACCEL option for the PROC COPY statement is enabled by default. This option controls whether data is copied in-database.

```

libname pgin postgres user="myuser" database=userdb port=5432
  server=pgcloud password=cloudpwd schema=mysch1;
libname pgout postgres user="myuser" database=userpg port=5432
  server=altpgsvr password=mypwd2 schema=mysch2;

proc copy in=pgin out=pgout;
  select proj_tab;
run;
quit;

proc print data=pgout.proj_tab; run;

```

## Passing SAS Functions to PostgreSQL

SAS/ACCESS Interface to PostgreSQL passes the following SAS functions to PostgreSQL for processing. Where the PostgreSQL function name differs from the SAS function name, the PostgreSQL name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                      |                   |
|----------------------|-------------------|
| ABS                  | LOWCASE (LOWER)   |
| ARCOS (ACOS)         | MAX               |
| ARSIN (ASIN)         | MIN               |
| ATAN                 | MINUTE            |
| AVG                  | MOD (see note)    |
| BYTE                 | MONTH             |
| CEIL                 | QTR               |
| COALESCE             | REPEAT            |
| COMPRESS (TRANSLATE) | SECOND            |
| COS                  | SIGN              |
| COT                  | SIN               |
| COUNT                | SQRT              |
| DAY                  | STD (STDDEV_SAMP) |

|                            |                      |
|----------------------------|----------------------|
| DAYOFWEEK                  | STRIP (BTRIM)        |
| EXP                        | SUBSTR               |
| FLOOR                      | SUM                  |
| HOUR                       | TAN                  |
| INDEX (POSITION, STRPOS)   | TRANSLATE (see note) |
| LENGTH                     | TRANWRD (REPLACE)    |
| LENGTHC (CHARACTER_LENGTH) | TRIMN (RTRIM)        |
| LENGTHN (LENGTH)           | UPCASE (UPPER)       |
| LOG (LN)                   | VAR (VAR_SAMP)       |
| LOG10 (LOG)                | YEAR                 |
| LOG2 (LOG)                 |                      |

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

**Note:** If more than three arguments are provided for the TRANSLATE function, then TRANSLATE is performed by SAS and the function is not passed to the database.

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to PostgreSQL. Due to incompatibility in date and time functions between PostgreSQL and SAS, PostgreSQL might not process them correctly. Check your results to determine whether these functions are working as expected.

|                     |                      |
|---------------------|----------------------|
| ATAN2               | ROUND                |
| DATE (current_date) | TIME (current_time)  |
| DATEPART (cast)     | TIMEPART (cast)      |
| DATETIME (now)      | TODAY (current_date) |
| MOD                 |                      |

## Passing Joins to PostgreSQL

For a multiple libref join to pass to PostgreSQL, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- database (DATABASE=)

- port (PORT=)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

## Bulk Loading and Unloading for PostgreSQL

### Overview of Bulk Loading and Unloading for PostgreSQL

Bulk loading is the fastest way to insert large numbers of rows into a PostgreSQL table. To use the bulk-load utility, set the [BULKLOAD=](#) data set option to YES. Similarly, bulk unloading is the fastest way to retrieve large numbers of rows from a PostgreSQL table. To use the bulk-unloading utility, set the [BULKUNLOAD=](#) data set option to YES. Bulk loading and unloading are implemented with the PostgreSQL PSQL utility, which moves data between SAS and the PostgreSQL database.

### Data Set Options with Bulk Loading and Unloading

Here are the PostgreSQL data set options for bulk loading and bulk unloading. For more information about these options, see “[Overview](#)” on page 370.

- [BL\\_DATAFILE=](#)
- [BL\\_DEFAULT\\_DIR=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DELIMITER=](#)
- [BL\\_ESCAPE=](#)
- [BL\\_FORMAT=](#)
- [BL\\_LOAD\\_METHOD=](#)

**Note:** This data set option applies to bulk loading only.

- [BL\\_LOGFILE=](#)
- [BL\\_NULL=](#)
- [BL\\_PSQL\\_PATH=](#)
- [BL\\_QUOTE=](#)

- [BULKLOAD=](#)
- [BULKUNLOAD=](#)

## Using the PSQL Tool for Bulk Loading and Unloading

Required for bulk loading and bulk unloading, the PSQL tool is a terminal-based front end to PostgreSQL. You can use it to enter queries interactively, submit them to PostgreSQL, and see the query results. You can also submit a file as input. PSQL provides a number of metacommands and various shell-like features to facilitate writing scripts and automating various tasks. It is available here: <http://www.postgresql.org>.

## Bulk-Load Example for PostgreSQL

This first example shows how you can use a SAS data set, SASFLT.FLT98, to create and load a large PostgreSQL table, FLIGHTS98:

```
libname sasflt 'SAS-library';
libname net_air postgres user=myusr1 pwd=mypwd1
      server=air2 database=flights;

proc sql;
  create table net_air.flights98 (bulkload=YES bl_psql_path='full-path-
of-PSQL')
    as select * from sasflt.flt98;
quit;
```

## Improve Performance When Deleting or Updating Rows

- 1 Ensure that your table has a primary key.
  - 2 Disable the DBIDIRECTEXEC system option by specifying the OPTIONS statement:
- ```
options nodbidirectexec;
```
- 3 Specify the CURSOR\_TYPE=FORWARD\_ONLY option in your LIBNAME statement:

```
libname delpg postgres server="myserverIP" db=myDatabase user=myUser
      pwd=myPWD cursor_type=forward_only;
```

You might need to enable the DBIDIRECTEXEC system option after you complete your updates. This system option enables other functions or procedures to run within the PostgreSQL database. To enable DBIDIRECTEXEC, specify this OPTIONS statement:

```
options dbidirectexec;
```

## Locking in the PostgreSQL Interface

The following LIBNAME and data set options let you control how the PostgreSQL interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134.

`READ_LOCK_TYPE= ROW`

`UPDATE_LOCK_TYPE= ROW`

`READ_ISOLATION_LEVEL= S | RC`

The PostgreSQL ODBC driver manager supports the S and RC isolation levels that are specified in this table.

*Table 33.3 Isolation Levels for PostgreSQL*

Isolation Level	Definition
S (serializable)	Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.
RR (repeatable read)	Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.
RC (read committed)	Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.
RU (read uncommitted)	Allows dirty Reads, nonrepeatable Reads, and phantom Reads,
V (versioning)	Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads. These transactions are serializable but higher concurrency is possible than with the serializable isolation level. Typically, a nonlocking protocol is used.

Here are how the terms in the table are specified.

### *Dirty reads*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it can see changes that are made by those concurrent transactions even before they commit.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

*Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist, a phantom.

`UPDATE_ISOLATION_LEVEL= S | RC`

The PostgreSQL ODBC driver manager supports the S and RC isolation levels that are specified in the preceding table.

## Working with NLS Characters in PostgreSQL Data

If your data contains NLS characters, you might receive errors in the SAS log that stop processing for a DATA step. If you prefer to continue with processing, you can enable the `SAS_POSTGRES_UPDATE_WARNING=` environment variable. When you specify `SAS_POSTGRES_UPDATE_WARNING=1`, then a warning is printed to the SAS log and processing continues.

The warning might resemble the following example:

```
WARNING: During update: [SAS] [ODBC PostgreSQL Wire Protocol driver]No rows were
affected by UPDATE/DELETE WHERE CURRENT OF cursor
emulation
```

## Naming Conventions for PostgreSQL

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The PostgreSQL interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The `PRESERVE_TAB_NAMES=` and `PRESERVE_COL_NAMES=` options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for](#)

([Relational Databases on page 127](#).) PostgreSQL is not case sensitive, and all names default to lowercase.

PostgreSQL objects include tables, views, and columns. They follow these naming conventions.

- A name can contain up to 128 characters.
- The first character in a name can be a letter, @, \_, or #.
- A name cannot be a PostgreSQL reserved word, such as WHERE or VIEW.
- A name cannot be the same as another PostgreSQL object that has the same type.
- A name must be unique within each type of each object.

---

## Data Types for PostgreSQL

---

### Overview of Data Types

Every column in a table has a name and a data type. The data type tells PostgreSQL how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about PostgreSQL data types, null and default values, and data conversions.

For more information about PostgreSQL data types and to determine which data types are available for your version of PostgreSQL, see your PostgreSQL documentation.

---

### PostgreSQL Null Values

PostgreSQL has a special value called NULL. A PostgreSQL NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a PostgreSQL NULL value, it interprets it as a SAS missing value.

You can define a column in a PostgreSQL table so that it requires data. To do this in SQL, you specify a column as NOT NULL. This tells SQL to allow a row to be added to a table only if a value exists for the field. For example, NOT NULL assigned to the CUSTOMER field in the SASDEMO.CUSTOMER table does not allow a row to be added unless there is a value for CUSTOMER. When creating a table with SAS/ACCESS, you can use the DBNULL= data set option to indicate whether NULL is a valid value for specified columns.

You can also specify PostgreSQL columns as NOT NULL DEFAULT. For more information about using the NOT NULL DEFAULT value, see your PostgreSQL documentation.

Once you know whether a PostgreSQL column enables NULLs or the host system provides a default value for a column that is specified as NOT NULL WITH DEFAULT, you can write selection criteria and enter values to update a table. Unless a column is specified as NOT NULL or NOT NULL DEFAULT, it allows NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how SAS missing character values are handled, use the **NULCHAR=** and **NULCHARVAL=** data set options.

## Working with Long Character Values in PostgreSQL

The PostgreSQL open-source driver automatically assigns a length of 255 characters to varying-length character variables, such as the VARCHAR(*n*) data type. Therefore, if you know that character data in a data set is longer than 255 characters, you should specify the MaxVarcharSize attribute for the CONOPTS=LIBNAME option when you connect to your PostgreSQL library. Specify a value for MaxVarcharSize that is as large as the longest column value to ensure that you retrieve all of the available data.

The following LIBNAME statement, sample tables, and SQL query would create a new data set that does not truncate the character data from the Testb data set.

```
libname x postgres preserve_col_names=no
        preserve_tab_names=no database=mydb1 dbmax_text=32767
        server='myserver' port=5432 user=pgadmin password='pgpwd'
        conopts='MaxVarcharSize=300';

/* Create example tables Testa and Testb */
proc sql;
create table testa(idnum int, myflag int, col1 varchar(20))
create table testb(idnum int, col1 varchar(300))

insert into testa values (1, 1, 'from a 1')
insert into testa values (2, 0, 'from a 2')

insert into testb values (1, 'from b 1*from b 1*from b 1*from b 1*from
b 1*
        from b 1*from b 1*from b 1*from b 1*from b 1*from b 1*from
b 1*
        from b 1*from b 1*from b 1*from b 1*from b 1*from b 1*from
b 1*
        from b 1*from b 1*from b 1*from b 1*from b 1*from b 1*from
b 1*
        from b 1*274')
insert into testb values (2, 'from b 2*from b 2*from b 2*from b 2*from
b 2*
        from b 2*from b 2*from b 2*from b 2*from b 2*from b 2*from
b 2*
        from b 2*from b 2*from b 2*from b 2*from b 2*from b 2*from
b 2*
        from b 2*from b 2*from b 2*from b 2*from b 2*from b 2*from
b 2*
        from b 2*274')
```

```

quit;

/* Verify that data is not truncated */
proc sql;
create table work.SASTEST as
  select case when myflag = 1 then a.col1 else b.col1 end as
    col1 from x.testa a, x.testb b
  where a.idnum = b.idnum;

/* Select the end of col1 from position 200 to the end to show that
all of */
/* the data is
retrieved
select substr(col1, 200) from work.SASTEST;
quit;

```

The output from the final SELECT statement appears as shown here. This statement selects a substring starting at column position 200 from the Col1 value.

```
rom b 2*from b 2*274
```

You can see that the end of the value does include the ‘\*274’ from the end of the VARCHAR value.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to PostgreSQL assigns to SAS variables when using the [LIBNAME statement](#) to read from a PostgreSQL table. These default formats are based on PostgreSQL column attributes. SAS/ACCESS does not support PostgreSQL data types that do not appear in this table.

*Table 33.4 LIBNAME Statement: Default SAS Formats for PostgreSQL Data Types*

PostgreSQL Data Type	ODBC Data Type	Default SAS Format
CHAR( <i>n</i> ) <sup>1</sup>	SQL_CHAR	\$ <i>w</i> .
VARCHAR( <i>n</i> ) <sup>1</sup>	SQL_LONGCHAR( <i>n</i> )	
BYTEA	SQL_VARBINARY	\$ <i>w</i> <sup>4</sup>
		SQL_LONGVARBINARY <sup>3</sup>
DECIMAL	SQL_DECIMAL	<i>w</i> or <i>w.d</i> or none if you do not specify <i>w</i> and <i>d</i> <sup>2</sup>
NUMERIC	SQL_NUMERIC	
INTEGER	SQL_INTEGER	11.
SMALLINT	SQL_SMALLINT	6.

PostgreSQL Data Type	ODBC Data Type	Default SAS Format
	SQL_TINYINT	4.
BIT(1)	SQL_BIT	1.
REAL	SQL_REAL	none
FLOAT	SQL_FLOAT	
DOUBLE PRECISION	SQL_DOUBLE	
BIGINT	SQL_BIGINT	20.
INTERVAL	SQL_INTERVAL	\$w.
UUID	SQL_GUID	\$w.
DATE	SQL_TYPE_DATE	DATE9.

1  $n$  in PostgreSQL character data types is equivalent to  $w$  in SAS formats.

2  $m$  and  $n$  in PostgreSQL numeric data types are equivalent to  $w$  and  $d$  in SAS formats.

3 This conversion occurs when you specify "ByteAsLongVarBinary=1" with the CONOPS= LIBNAME option.

4 Because the Postgres ODBC driver does the actual conversion, this field is displayed as if the \$HEXw. format were applied.

The following table shows the default data types that SAS/ACCESS Interface to PostgreSQL uses when creating tables. The PostgreSQL engine lets you specify nondefault data types by using the DBTYPE= data set option.

**Table 33.5 LIBNAME Statement: Default SAS Formats for PostgreSQL Data Types When Creating Tables**

PostgreSQL Data Type	ODBC Data Type	Default SAS Format
NUMERIC( $m,n$ ) <sup>2</sup>	SQL_DOUBLE or SQL_NUMERIC using $m,n$ if the DBMS allows it	$w.d$
VARCHAR( $n$ ) <sup>1</sup>	SQL_VARCHAR using $n$	\$w.
TIMESTAMP <sup>3</sup>	SQL_TIMESTAMP	DATETIME formats
DATE	SQL_DATE	DATE formats
TIME	SQL_TIME	TIME formats

1  $n$  in PostgreSQL character data types is equivalent to  $w$  in SAS formats.

2  $m$  and  $n$  in PostgreSQL numeric data types are equivalent to  $w$  and  $d$  in SAS formats.

3 Although the PostgreSQL engine supports TIMESTAMP, it has no TIMESTAMP WITH TIMEZONE data type that maps to the corresponding Postgres data type.

---

# Sample Programs for PostgreSQL

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.

# SAS/ACCESS Interface to SAP ASE

---

<i>System Requirements for SAS/ACCESS Interface to SAP ASE</i> .....	1196
<i>Introduction to SAS/ACCESS Interface to SAP ASE</i> .....	1196
<b><i>LIBNAME Statement for the SAP ASE Engine</i></b> .....	1197
Overview .....	1197
Arguments .....	1197
SAP ASE LIBNAME Statement Example .....	1201
<b><i>Data Set Options for SAP ASE</i></b> .....	1201
<b><i>SQL Pass-Through Facility Specifics for SAP ASE</i></b> .....	1203
Key Information .....	1203
SQL Example for SAP ASE .....	1203
<b><i>Autopartitioning Scheme for SAP ASE</i></b> .....	1204
Overview .....	1204
Indexes .....	1204
Partitioning Criteria .....	1205
Data Types .....	1205
Autopartitioning Examples .....	1205
<b><i>Temporary Table Support for SAP ASE</i></b> .....	1206
<b><i>ACCESS Procedure Specifics for SAP ASE</i></b> .....	1206
Overview .....	1206
PROC ACCESS Example for SAP ASE .....	1206
<b><i>DBLOAD Procedure Specifics for SAP ASE</i></b> .....	1208
Overview .....	1208
PROC DBLOAD Example for SAP ASE .....	1209
<b><i>Passing SAS Functions to SAP ASE</i></b> .....	1210
<b><i>Passing Joins to SAP ASE</i></b> .....	1210
<b><i>Bulk Loading for SAP ASE</i></b> .....	1211
Overview .....	1211
Data Set Options for Bulk Loading .....	1212

<i>Reading Multiple SAP ASE Tables</i>	1212
<i>Locking in the SAP ASE Interface</i>	1213
Overview	1213
Understanding SAP ASE Update Rules	1214
<i>Naming Conventions for SAP ASE</i>	1214
<i>Case Sensitivity in SAP ASE</i>	1215
<i>Data Types for SAP ASE</i>	1216
Overview	1216
Supported SAP ASE Data Types	1216
User-Defined Data	1217
SAP ASE Null Values	1217
LIBNAME Statement Data Conversions	1218
ACCESS Procedure Data Conversions	1219
DBLOAD Procedure Data Conversions	1220
Data Returned as SAS Binary Data with Default Format \$HEX	1221
Data Returned as SAS Character Data	1221
Inserting TEXT into SAP ASE from SAS	1222
<i>National Language Support for SAP ASE</i>	1222

---

## System Requirements for SAS/ACCESS Interface to SAP ASE

You can find information about system requirements for SAS/ACCESS Interface to SAP ASE in the following locations:

- [System Requirements for SAS/ACCESS Interface to SAP ASE with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

## Introduction to SAS/ACCESS Interface to SAP ASE

For available SAS/ACCESS features, see [SAP ASE supported features on page 114](#). For more information about SAP ASE, see your SAP ASE documentation.

---

**Note:** In [SAS 9.4M4](#), the SAS/ACCESS product name changed from SAS/ACCESS Interface to Sybase to SAS/ACCESS Interface to SAP ASE to reflect the name change of the SAP ASE product.

---

# LIBNAME Statement for the SAP ASE Engine

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to SAP ASE supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing SAP ASE.

**LIBNAME** *libref* **sapase** <connection-options> <LIBNAME-options>;

**Note:** The previous engine name, sybase, is also accepted for this SAS/ACCESS engine.

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

## Arguments

### *libref*

any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### **sapase**

the SAS/ACCESS engine name for the SAP ASE interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here are the connection options for SAP ASE. All options are all case sensitive: They are passed to SAP ASE exactly as you enter them.

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

### USER=<'>SAP-ASE-user-name<'>

specifies the SAP ASE user name (also called the login name) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

PASSWORD=<'>SAP-ASE-password<'>

specifies the password that is associated with the SAP ASE user name. If you omit the password, a default password of NULL is used. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: PASS=, PW=, SYBPW=

DATABASE=<'>SAP-ASE-database-name<'>

specifies the name of the SAP ASE database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. If you omit DATABASE=, the default database for your SAP ASE user name is used.

Alias: DB=

SERVER=<'>server-name<'>

specifies the server that you want to connect to. This server accesses the database that contains the tables and views that you want to access. If the server name contains lowercase, spaces, or nonalphanumeric characters, you must enclose it in quotation marks. If you omit SERVER=, the default action for your operating system occurs. On UNIX systems, the value of the environment variable DSQUERY is used if it has been specified.

INTERFACE=*file-name*

specifies the name and location of the SAP ASE interfaces file. This file contains the names and network addresses of all available servers on the network. If you omit this statement, the default action for your operating system occurs. INTERFACE= is not used in some operating environments. Contact your database administrator to determine whether it applies to your operating environment.

IP\_CURSOR= YES | NO

specifies whether implicit PROC SQL pass-through processes multiple result data sets simultaneously. IP\_CURSOR is set to NO by default. Setting it to YES allows this type of extended processing. However, it decreases performance because cursors, not result data sets, are being used. Do not set to YES unless needed.

SYBBUFSZ=*number-of-rows*

specifies the number of rows of DBMS data to write to the buffer. If this statement is used, the SAS/ACCESS interface view engine creates a buffer that is large enough to hold the specified number of rows. This buffer is created when the associated database table is read. The interface view engine uses SYBBUFSZ= to improve performance. If you omit this statement, no data is written to the buffer.

If you specify the appropriate system options or environment variables for your database, you can often omit the connection options. See your SAP ASE documentation for details.

*LIBNAME-options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to SAP ASE, with the applicable default values. This table also identifies LIBNAME options that, when specified in a LIBNAME statement, are used in the CONNECT statement in the SQL procedure. (See column Valid in CONNECT.) For details, see [LIBNAME Options for Relational Databases on page 134..](#)

**Table 34.1** SAS/ACCESS LIBNAME Options for SAP ASE

Option	Default Value	Valid in CONNECT
ACCESS=	none	
AUTHDOMAIN=	none	•
AUTOCOMMIT=	YES	•
CONNECTION=	UNIQUE when data source supports only one cursor per connection; otherwise, SHAREDREAD	•
CONNECTION_GROUP=	none	•
DBCOMMIT=	1000 (when inserting rows), 0 (when updating rows)	
DBCONINIT=	none	•
DBCONTERM=	none	•
DBCREATE_TABLE_OPTS=	none	
DBGEN_NAME=	DBMS	•
DBINDEX=	NO	
DBLIBINIT=	none	
DBLIBTERM=	none	
DBLINK=	the local database	•
DBMAX_TEXT=	1024	•
DBPROMPT=	NO	•
DBSASLABEL=	COMPAT	
DBSERVER_MAX_BYTES=	usually 1	•
DBSLICEPARM=	THREADED_APPS,2 or THREADED_APPS,3	
DEFER=	NO	•

Option	Default Value	Valid in CONNECT
DIRECT_EXE=	none	
DIRECT_SQL=	YES	
ENABLE_BULK=	YES	
INTERFACE=	none	•
MAX_CONNECTS=	25	•
MULTI_DATASRC_OPT=	NONE	
POST_STMT_OPTS=	none	
PACKETSIZE=	server value	•
QUOTED_IDENTIFIER=	NO	
READBUFF=	100	•
READ_ISOLATION_LEVEL=	1 (see “Locking in the SAP ASE Interface”)	•
READ_LOCK_TYPE=	NOLOCK (see “Locking in the SAP ASE Interface”)	•
REREAD_EXPOSURE=	NO	
SCHEMA=	none	
SPOOL=	YES	
SQL_FUNCTIONS=	none	
SQL_FUNCTIONS_COPY=	none	
SQL_OJ_ANSI=	NO	
UPDATE_ISOLATION_LEVEL=	1 (see “Locking in the SAP ASE Interface”)	
UPDATE_LOCK_TYPE=	PAGE (see “Locking in the SAP ASE Interface”)	
UTILCONN_TRANSIENT=	NO	

---

## SAP ASE LIBNAME Statement Example

In the following example, the libref MYDBLIB uses the SAP ASE engine to connect to an SAP ASE database. USER= and PASSWORD= are connection options.

```
libname mydblib sapase user=myusr1 password=mypwd1;
```

If you specify the appropriate system options or environment variables for your database, you can often omit the connection options. See your SAP ASE documentation for details.

---

## Data Set Options for SAP ASE

All SAS/ACCESS data set options in this table are supported for SAP ASE. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 34.2** SAS/ACCESS Data Set Options for SAP ASE

Option	Default Value
AUTOCOMMIT=	LIBNAME option value
BULK_BUFFER=	100
BULKLOAD=	NO
DBCOMMIT=	LIBNAME option value
DBCONDITION=	none
DBCREATE_TABLE_OPTS=	LIBNAME option value
DBFORCE=	NO
DBGEN_NAME=	LIBNAME option value
DBINDEX=	LIBNAME option value
DBKEY=	none
DBLABEL=	NO
DBLINK=	LIBNAME option value

Option	Default Value
DBLARGETABLE=	none
DBMAX_TEXT=	1024
DBNULL=	_ALL_=YES
DBPROMPT=	LIBNAME option value
DBSASLABEL=	COMPAT
DBSLICE=	none
DBSLICEPARM=	THREADED_APPS,2 or THREADED_APPS,3
DBTYPE=	see “Data Types for SAP ASE” on page 1216
ERRLIMIT=	1
NULLCHAR=	SAS
NULLCHARVAL=	a blank character
POST_STMT_OPTS=	none
POST_TABLE_OPTS=	none
PRE_STMT_OPTS=	none
PRE_TABLE_OPTS=	none
READBUFF=	LIBNAME option value
READ_ISOLATION_LEVEL=	LIBNAME option value
READ_LOCK_TYPE=	LIBNAME option value
SASDATEFMT=	DATETIME22.3
SCHEMA=	LIBNAME option value
SEGMENT_NAME=	none
UPDATE_ISOLATION_LEVEL=	LIBNAME option value
UPDATE_LOCK_TYPE=	LIBNAME option value

---

# SQL Pass-Through Facility Specifics for SAP ASE

---

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the SAP ASE interface.

- The *dbms-name* is **SAPASE**.
- The CONNECT statement is optional. If you omit the CONNECT statement, an implicit connection is made using the default values for all connection options.
- The interface can connect multiple times to one or more servers.
- The *database-connection-arguments* for the CONNECT statement are identical to its LIBNAME [connection options](#).
- These LIBNAME options are also available with the CONNECT statement.
  - [DBMAX\\_TEXT=](#)
  - [MAX\\_CONNECTS=](#)
  - [READBUFF=](#)
  - [PACKETSIZE=](#)

---

## SQL Example for SAP ASE

This example retrieves a subset of rows from the SAP ASE INVOICE table. Because the WHERE clause is specified in the DBMS query (the inner SELECT statement), the DBMS processes the WHERE expression and returns a subset of rows to SAS.

```
proc sql;
connect to sapase(server=MYSRV1 database=INVENTORY
                   user=myusr1 password=mypwd1);
%put &sqlxmsg;

select * from connection to sapase
  (select * from INVOICE where BILLEDBY=457232);
%put &sqlxmsg;
```

The SELECT statement that is enclosed in parentheses is sent directly to the database and therefore must be specified using valid database variable names and syntax.

---

# Autopartitioning Scheme for SAP ASE

---

## Overview

For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

SAP ASE autopartitioning uses the SAP ASE MOD function (%) to create multiple SELECT statements with WHERE clauses. In the optimum scenario, the WHERE clauses divide the result set into equal chunks: one chunk per thread. For example, assume that your original SQL statement was `SELECT * FROM DBTAB`, and assume that DBTAB has a primary key column PKCOL of type integer and that you want it partitioned into three threads. Here is how the autopartitioning scheme would break up the table into three SQL statements:

```
select * from DBTAB where (abs(PKCOL))%3=0
select * from DBTAB where (abs(PKCOL))%3=1
select * from DBTAB where (abs(PKCOL))%3=2
```

Because PKCOL is a primary key column, you should receive a fairly even distribution among the three partitions, which is the primary goal.

---

## Indexes

An index on a SAS partitioning column increases performance of the threaded Read. If a primary key is not specified for the table, an index should be placed on the partitioning column in order to attain similar benefits. To achieve optimum database performance, it is essential to understand and follow the recommendations in the *SAP ASE Performance and Tuning Guide* for creating and using indexes. Here is the order of column selection for the partitioning column.

- 1 Identity column
- 2 Primary key column (INTEGER or NUMERIC)
- 3 INTEGER, NUMERIC, or BIT; not nullable
- 4 INTEGER, NUMERIC, or BIT; nullable

If the column selected is a bit type, only two partitions are created because the only values are 0 and 1.

---

## Partitioning Criteria

The most efficient partitioning column is an Identity column, which is usually identified as a primary key column. Identity columns usually lead to evenly partitioned result data sets because of the sequential values that they store.

The least efficient partitioning column is a numeric, decimal, or float column that is **NULLABLE** and that does not have a specified index.

Given equivalent selection criteria, columns specified at the beginning of the table definition that meet the selection criteria takes precedence over columns specified toward the end of the table definition.

---

## Data Types

These data types are supported in partitioning column selection:

- BIT
- DECIMAL
- FLOAT
- INTEGER
- NUMERIC
- SMALLINT
- TINYINT

---

## Autopartitioning Examples

Here are examples of generated SELECT statements involving various column data types.

COL1 is NUMERIC, DECIMAL, or FLOAT. This example uses three threads (the default) and COL1 is NOT NULL.

```
select * from DBTAB where (abs(convert(INTEGER, COL1)))%3=0  
select * from DBTAB where (abs(convert(INTEGER, COL1)))%3=1  
select * from DBTAB where (abs(convert(INTEGER, COL1)))%3=2
```

COL1 is BIT, INTEGER, SMALLINT, or TINYINT. This example uses two threads (the default) and COL1 is NOT NULL.

```
select * from DBTAB where (abs(COL1))%3=0  
select * from DBTAB where (abs(COL1))%3=1
```

COL1 is an INTEGER and is nullable.

```
select * from DBTAB where (abs(COL1))%3=0 OR COL1 IS NULL  
select * from DBTAB where (abs(COL1))%3=1
```

# Temporary Table Support for SAP ASE

SAS/ACCESS Interface to SAP ASE supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## ACCESS Procedure Specifics for SAP ASE

### Overview

For general information about this feature, see the [Appendix 1, “ACCESS Procedure,” on page 1421](#).

SAS/ACCESS for SAP ASE supports all [ACCESS procedure statements](#).

Here are the ACCESS Procedure specifics for SAP ASE.

**Note:** SAS still supports this legacy procedure. However, to access your DBMS data more directly the best practice is to use the LIBNAME statement for SAP ASE or the SQL pass-through facility. For more information, see “[LIBNAME Statement for the SAP ASE Engine](#)” on page 1197 and “[SQL Pass-Through Facility Specifics for SAP ASE](#)” on page 1203.

- The DBMS= value for PROC ACCESS is SAPASE.
- The *database-description-statements* that PROC ACCESS uses are identical to the [database-connection-arguments](#) in the CONNECT statement for the SQL pass-through facility.
- Here is the TABLE= statement for PROC ACCESS.

TABLE= '>*table-name*<'>;  
specifies the name of the SAP ASE table or SAP ASE view on which the access descriptor is based.

### PROC ACCESS Example for SAP ASE

The following example creates access descriptors and view descriptors for the EMPLOYEES and INVOICE tables. These tables have different owners and are

stored in PERSONNEL and INVENTORY databases that reside on different machines. The USER= and PASSWORD= statements identify the owners of the SAP ASE tables and their passwords.

```

libname vlib 'sas-library';

proc access dbms=sapase;
  create work.employee.access;
    server='mysrv1'; database='personnel';
    user='myusr1'; password='mypwd1';
    table=EMPLOYEES;
  create vlib.emp_acc.view;
    select all;
    format empid 6. ;
    subset where DEPT like 'ACC%';
run;

proc access dbms=sapase;
  create work.invoice.access;
    server='mysrv2'; database='inventory';
    user='myusr2'; password='mypwd2';
    table=INVOICE;
    rename invocenum=invnum;
    format invocenum 6. billedon date9.
      paidon date9. ;
  create vlib.sainv.view;
    select all;
    subset where COUNTRY in ('Argentina','Brazil');
run;

options linesize=120;
title 'South American Invoices and
      Who Submitted Them';

proc sql;
  select invnum, country, billedon, paidon,
         billedby, lastname, firstnam
  from vlib.emp_acc, vlib.sainv
  where emp_acc.empid=sainv.billedby;

```

SAP ASE is a case-sensitive database. The PROC ACCESS database identification statements and the SAP ASE column names in all statements except SUBSET are converted to uppercase unless the names are enclosed in quotation marks. The SUBSET statements are passed to SAP ASE exactly as you enter them, so you must use the correct case for the SAP ASE column names.

# DBLOAD Procedure Specifics for SAP ASE

## Overview

For general information about this feature, see the [Appendix 3, “DBLOAD Procedure,” on page 1455](#).

The SAP ASE interface supports all **DBLOAD** procedure statements.

**Note:** SAS still supports this legacy procedure. However, to send data to your DBMS more directly the best practice is to use the LIBNAME statement for SAP ASE or the SQL pass-through facility. For more information, see [“LIBNAME Statement for the SAP ASE Engine” on page 1197](#) and [“SQL Pass-Through Facility Specifics for SAP ASE” on page 1203](#).

Here are the SAP ASE interface specifics for the DBLOAD procedure.

- The DBMS= value for PROC DBLOAD is SAPASE.

- The TABLE= statement for PROC DBLOAD is:

TABLE= '>*table-name*<'>;

- PROC DBLOAD uses these *database-description-statements*.

USER=<'>SAP-ASE-user-name<'>

specifies the SAP ASE user name (also called the login name) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

PASSWORD=<'>SAP-ASE-password<'>

specifies the password that is associated with the SAP ASE user name.

If you omit the password, a default password of NULL is used. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

PASSWORD= can also be specified with the SYBPW=, PASS=, and PW= aliases.

DATABASE=<'>*database-name*<'>

specifies the name of the SAP ASE database that contains the tables and views that you want to access.

If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. If you omit DATABASE=, the default database for your SAP ASE user name is used.

Alias: DB=

**SERVER=<'>server-name<'>**

specifies the server that you want to connect to. This server accesses the database that contains the tables and views that you want to access.

If the server name contains lowercase, spaces, or nonalphanumeric characters, you must enclose it in quotation marks.

If you omit SERVER=, the default action for your operating system occurs. On UNIX systems, the value of the environment variable DSQUERY is used if it has been specified.

**INTERFACE=file-name**

specifies the name and location of the SAP ASE interfaces file. The interfaces file contains the names and network addresses of all available servers on the network.

If you omit this statement, the default action for your operating system occurs. INTERFACE= is not used in some operating environments. Contact your database administrator to determine whether it applies to your operating environment.

**BULKCOPY=Y|N;**

uses the SAP ASE bulk copy utility to insert rows into an SAP ASE table. The default value is N.

If you specify BULKCOPY=Y, BULKCOPY= calls the SAP ASE bulk copy utility in order to load data into an SAP ASE table. This utility groups rows so that they are inserted as a unit into the new table. Using the bulk copy utility can improve performance.

You use the COMMIT= statement to specify the number of rows in each group (this argument must be a positive integer). After each group of rows is inserted, the rows are permanently saved in the table. As each group is being inserted, if one row in the group is rejected, all rows in that group are rejected.

If you specify BULKCOPY=N, rows are inserted into the new table using Transact-SQL INSERT statements. See your SAP ASE documentation for more information about the bulk copy utility.

## PROC DBLOAD Example for SAP ASE

The following example creates a new SAP ASE table, EXCHANGE, from the DLIB.RATEOFEX data file. (The DLIB.RATEOFEX data set is included in the sample data that is shipped with your software.) An access descriptor ADLIB.EXCHANGE is also created, and it is based on the new table. The DBLOAD procedure sends a Transact-SQL GRANT statement to SAP ASE. You must be granted SAP ASE privileges to create new SAP ASE tables or to grant privileges to other users.

```
libname adlib 'SAS-library';
libname dlib 'SAS-library';

proc dbload dbms=sapase data=dlib.rateofex;
  server='mysrv1'; database='testdb'; user='myusr1';
  password='mypwd1'; table=EXCHANGE; accdesc=adlib.exchange;
  rename fgnindol=fgnindolar 4=dolrsinfgn;
```

```

nulls updated=n fgnindol=n 4=n country=n;
load;
run;

```

## Passing SAS Functions to SAP ASE

SAS/ACCESS Interface to SAP ASE passes the following SAS functions to SAP ASE for processing if the DBMS driver or client that you are using supports the function. Where the SAP ASE function name differs from the SAS function name, the SAP ASE name appears in parentheses. For information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

ABS	MAX
ARCOS (ACOS)	MIN
ARSIN (ASIN)	MINUTE
ATAN	MONTH
AVG	SECOND
CEIL (CEILING)	SIGN
COS	SIN
COUNT	SQRT
DATETIME (GETDATE())	STRIP (RTRIM(LTRIM))
DAY	SUM
EXP	TAN
FLOOR	TRIMN (RTRIM)
HOUR	UPCASE (UPPER)
LOG	WEEKDAY
LOWCASE (LOWER)	YEAR

`SQL_FUNCTIONS=ALL` allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when `SQL_FUNCTIONS=ALL` can the SAS/ACCESS engine also pass these SAS SQL functions to SAP ASE. Due to incompatibility in date and time functions between SAP ASE and SAS, SAP ASE might not process them correctly. Check your results to determine whether these functions are working as expected.

DATEPART	TIMEPART
ROUND	

## Passing Joins to SAP ASE

For a multiple libref join to pass to SAP ASE, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- database (DATABASE=)
- server (SERVER=)

SAP ASE supports nonstandard outer-join syntax. Outer joins between two or more tables can be passed to SQP ASE with these restrictions:

- Full outer joins are not supported.
- Only the '=' comparison operator is allowed in an ON clause.

For all queries that use an ON clause, both operands in the ON clause must reference a column name. Literal operands cannot be passed down to the DBMS. Because literal operands are not supported, all ON clauses are transformed into WHERE clauses before passing joins down to the DBMS. This can result in a query not being passed down if it contains additional WHERE clauses or if it contains complex join conditions.

SAP ASE evaluates multijoins that have WHERE clauses differently than SAS. Therefore, instead of passing multiple joins or joins that have additional WHERE clauses to SAP ASE, use the DIRECT\_SQL= LIBNAME option to let PROC SQL process the join internally.

For more information about when and how SAS/ACCESS passes joins to the DBMS, see [“Passing Joins to the DBMS” on page 61](#).

---

# Bulk Loading for SAP ASE

---

## Overview

Bulk loading is the fastest way to insert large numbers of rows into an SAP ASE table. To use the bulk-load facility, specify **BULKLOAD=YES**. The bulk-load facility uses the SAP ASE bulk copy facility to move data from SAS to SAP ASE. See the [ENABLE\\_BULK= LIBNAME option on page 240](#).

When **BULKLOAD=NO**, insertions are processed and rolled back as expected according to **DBCOMMIT=** and **ERRLIMIT=** values. If the **ERRLIMIT=** value is encountered, all uncommitted rows are rolled back. The **DBCOMMIT=** data set option determines the commit intervals.

When **BULKLOAD=YES**, the first error encountered causes the remaining rows—including the erroneous row—in the buffer to be rejected. No other errors within the same buffer are detected, even if the **ERRLIMIT=** value is greater than one. In addition, all rows before the error are committed, even if **DBCOMMIT=** is larger than the number of the erroneous row.

## Data Set Options for Bulk Loading

Here are the SAP ASE bulk-load data set options. For detailed information about these options, see “[Data Set Options](#)” on page 365.

- [BULK\\_BUFFER=](#)
- [BULKLOAD=](#)

## Reading Multiple SAP ASE Tables

SAS opens multiple SAP ASE tables for simultaneous reading in these situations:

- When you are using PROC COMPARE. Here is an example:
- ```
proc compare base=syb.data1 compare=syb.data2;
```
- When you are running an SCL program that reads from more than one SAP ASE table simultaneously.
  - When you are joining SAP ASE tables in SAS—namely, when implicit pass-through is not used (DIRECT\_SQL=NO). Here are four examples:

```
proc sql ;
    select * from syb.table1, syb.table2 where table1.x=table2.x;

proc sql;
    select * from syb.table1 where table1.x = (select x from syb.table2
    where y = 33);

proc sql;
    select empname from syb.employee where empyears > all (select empyears
    from syb.employee where emptitle = 'salesrep');

proc sql ;
    create view myview as
        select * from employee where empyears > all (select empyears from
        syb.employee where emptitle = 'salesrep');
quit;

proc print data=myview;
run;
```

To read two or more SAP ASE tables simultaneously, you must specify either the LIBNAME option CONNECTION=UNIQUE or the LIBNAME option READLOCK\_TYPE=PAGE. Because READLOCK\_TYPE=PAGE can degrade performance, it is generally recommended that you use CONNECTION=UNIQUE (unless there is a concern about the number of connections that are opened on the database).

---

# Locking in the SAP ASE Interface

---

## Overview

These LIBNAME and data set options let you control how the SAP ASE interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134.

**READ\_LOCK\_TYPE= PAGE | NOLOCK**

The default value for SAP ASE is NOLOCK.

**UPDATE\_LOCK\_TYPE= PAGE | NOLOCK**

**PAGE**

SAS/ACCESS uses a cursor that you can update. PAGE is the default value for SAP ASE. When you use this value, you cannot use the SCHEMA= option, and it is also recommended that the table have a specified primary key.

**NOLOCK**

SAS/ACCESS uses SAP ASE Browse mode updating, in which the table that is being updated must have a primary key and timestamp.

**READ\_ISOLATION\_LEVEL= 1 | 2 | 3**

For reads, SAP ASE supports isolation levels 1, 2, and 3, as specified in the following table. See your SAP ASE documentation for more information.

*Table 34.3 Isolation Levels for SAP ASE*

| Isolation Level | Definition                                                             |
|-----------------|------------------------------------------------------------------------|
| 1               | Prevents dirty Reads. This is the default transaction isolation level. |
| 2               | Uses serialized Reads.                                                 |
| 3               | Also uses serialized Reads.                                            |

**UPDATE\_ISOLATION\_LEVEL= 1 | 3**

SAP ASE uses a shared or update lock on base table pages that contain rows representing a current cursor position. This option applies to updates only when UPDATE\_LOCK\_TYPE=PAGE because cursor updating is in effect. It does not apply when UPDATE\_LOCK\_TYPE=NOLOCK.

For updates, SAP ASE supports isolation levels 1 and 3, as specified in the preceding table. See your SAP ASE documentation for more information.

---

## Understanding SAP ASE Update Rules

To avoid data integrity problems when updating and deleting data in SAP ASE tables, take these precautionary measures:

- Always specify a primary key.
- If the updates are not taking place through cursor processing, specify a timestamp column.

It is not always obvious whether updates are using cursor processing. Cursor processing is *never* used for LIBNAME statement updates if UPDATE\_LOCK\_TYPE=NOLOCK. Cursor processing is *always* used in these situations:

- Updates using the LIBNAME statement with UPDATE\_LOCK\_TYPE=PAGE. This is the default value for this option.
- Updates using PROC SQL views.
- Updates using PROC ACCESS view descriptors.

---

## Naming Conventions for SAP ASE

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

SAP ASE database objects include tables, views, columns, indexes, and database procedures. They follow these naming conventions.

- A name must be from 1 to 30 characters long—or 28 characters, if you enclose the name in quotation marks.
- A name must begin with an alphabetic character (A to Z) or an underscore (\_) unless you enclose the name in quotation marks.
- After the first character, a name can contain letters (A to Z) in uppercase or lowercase, numbers from 0 to 9, underscore (\_), dollar sign (\$), number sign (#), at sign (@), yen sign (¥), and monetary pound sign (£).
- Embedded spaces are not allowed unless you enclose the name in quotation marks.
- Embedded quotation marks are not allowed.
- Case sensitivity is specified when a server is installed. By default, the names of database objects are case sensitive. For example, the names CUSTOMER and customer are different on a case-sensitive server.
- By default, SAP ASE does not enclose column names and table names in quotations marks. To enclose these in quotation marks, you must use the [QUOTED\\_IDENTIFIER= LIBNAME](#) option when you assign a libref.

- When you use the DATASETS procedure to list your SAP ASE tables, the table names appear exactly as they exist in the SAP ASE data dictionary. If you specified the **SCHEMA= LIBNAME** option, SAS/ACCESS lists the tables for the specified schema user name.
- To reference a table or other named object that you own, or for the specified schema, use the table name (for example, CUSTOMERS). If you use the **DBLINK= LIBNAME** option, all references to the libref refer to the specified database.
- A name cannot be a reserved word in SAP ASE unless the name is enclosed in quotation marks. See your SAP ASE documentation for more information about reserved words.
- Database names must be unique. For each owner within a database, names of database objects must be unique. Column names and index names must be unique within a table.

---

## Case Sensitivity in SAP ASE

SAS names can be entered in either uppercase or lowercase. When you reference SAP ASE objects through the SAS/ACCESS interface, objects are case sensitive and require no quotation marks.

However, SAP ASE is generally specified for case sensitivity. Give special consideration to the names of such objects as tables and columns when the SAS ACCESS or DBLOAD procedures are to use them. The ACCESS procedure converts SAP ASE object names to uppercase unless they are enclosed in quotation marks. Any SAP ASE objects that were given lowercase names, or whose names contain national or special characters, must be enclosed in quotation marks. The only exceptions are the SUBSET statement in the ACCESS procedure and the SQL statement in the DBLOAD procedure. Arguments or values from these statements are passed to SAP ASE exactly as you enter them, with the case preserved.

In the SQL pass-through facility, all SAP ASE object names are case sensitive. The names are passed to SAP ASE exactly as they are entered.

For more information about case sensitivity and SAP ASE names, see “[Naming Conventions for SAP ASE](#)” on page [1214](#).

# Data Types for SAP ASE

## Overview

Every column in a table has a name and a data type. The data type indicates to the DBMS how much physical storage to reserve for the column and the format in which the data is stored. This section includes information about SAP ASE data types, null values, and data conversions, and also explains how to insert text into SAP ASE from SAS.

SAS/ACCESS does not support these SAP ASE data types: BINARY, VARBINARY, IMAGE, NCHAR( $n$ ), and NVARCHAR( $n$ ). SAS/ACCESS provides an error message when it tries to read a table that has at least one column that uses an unsupported data type.

## Supported SAP ASE Data Types

Here are the supported SAP ASE data types for SAS/ACCESS Interface to SAP ASE:

- Character data:

|                |      |
|----------------|------|
| CHAR( $n$ )    | TEXT |
| VARCHAR( $n$ ) |      |

**Note:** You must enclose all character data in single or double quotation marks.

- Numeric data:

|                  |          |
|------------------|----------|
| NUMERIC( $p,s$ ) | TINYINT  |
| DECIMAL( $p,s$ ) | SMALLINT |
| REAL             | INT      |
| FLOAT            | BIT      |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Date, time, and money data:

|               |            |
|---------------|------------|
| DATE          | TIMESTAMP  |
| TIME          | SMALLMONEY |
| SMALLDATETIME | MONEY      |
| DATETIME      |            |

## User-Defined Data

You can supplement the SAP ASE system data types by specifying your own data types with the SAP ASE system procedure `sp_addtype`. When you specify your own data type for a column, you can specify a default value (other than NULL) for it and specify a range of allowable values for it.

## SAP ASE Null Values

SAP ASE has a special value that is called NULL. A This value indicates an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads an SAP ASE NULL value, it interprets it as a SAS missing value.

By default, SAP ASE columns are specified as NOT NULL. NOT NULL tells SAP ASE not to add a row to the table unless the row has a value for the specified column.

If you want a column to accept NULL values, you must explicitly specify it as NULL. Here is an example of a CREATE TABLE statement that specifies all table columns as NULL except CUSTOMER. In this case, SAP ASE accepts a row only if it contains a value for CUSTOMER.

```
create table CUSTOMERS
  (CUSTOMER      char(8)      not null,
   STATE         char(2)       null,
   ZIPCODE       char(5)       null,
   COUNTRY       char(20)      null,
   TELEPHONE    char(12)      null,
   NAME          char(60)      null,
   CONTACT       char(30)      null,
   STREETADDRESS char(40)      null,
   CITY          char(25)      null,
   FIRSTORDERDATE datetime     null);
```

When you create an SAP ASE table with SAS/ACCESS, you can use the `DBNULL=` data set option to indicate whether NULL is a valid value for specified columns.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how SAS missing character values are handled, use the `NULCHAR=` and `NULLCHARVAL=` data set options.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to SAP ASE assigns to SAS variables when using the **LIBNAME statement** to read from an SAP ASE table. These default formats are based on SAP ASE column attributes.

**Table 34.4 LIBNAME Statement: Default SAS Formats for SAP ASE Server Data Types**

| SAP ASE Column Type | SAS Data Type | Default SAS Format                                                                    |
|---------------------|---------------|---------------------------------------------------------------------------------------|
| CHAR( <i>n</i> )    | character     | \$ <i>n</i> <sup>2</sup>                                                              |
| VARCHAR( <i>n</i> ) | character     | \$ <i>n</i> <sup>2</sup>                                                              |
| TEXT                | character     | \$ <i>n</i> . <sup>2</sup><br>(where <i>n</i> is the value of the DBMAX_TEXT= option) |
| BIT                 | numeric       | 1.0                                                                                   |
| TINYINT             | numeric       | 4.0                                                                                   |
| SMALLINT            | numeric       | 6.0                                                                                   |
| INT                 | numeric       | 11.0                                                                                  |
| NUMERIC             | numeric       | <i>w, w.d</i> (if possible)                                                           |
| DECIMAL             | numeric       | <i>w, w.d</i> (if possible)                                                           |
| FLOAT               | numeric       | BEST22.                                                                               |
| REAL                | numeric       | BEST11.                                                                               |
| SMALLMONEY          | numeric       | DOLLAR12.2                                                                            |
| MONEY               | numeric       | DOLLAR24.2                                                                            |
| DATE <sup>1</sup>   | numeric       | DATE9.                                                                                |
| TIME <sup>1</sup>   | numeric       | TIME12.                                                                               |
| SMALLDATETIME       | numeric       | DATETIME22.3                                                                          |
| DATETIME            | numeric       | DATETIME22.3                                                                          |

| SAP ASE Column Type | SAS Data Type | Default SAS Format |
|---------------------|---------------|--------------------|
| TIMESTAMP           | hexadecimal   | \$HEXw             |

- 1 If a conflict might occur between the SAP ASE and SAS value for this data type, use `SASDATEFMT=` to specify the SAS format.  
 2 *n* specifies the current value for the **Adaptive Server** page size.

This table shows the default SAP ASE data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 34.5 LIBNAME STATEMENT: Default SAP ASE Data Types for SAS Variable Formats**

| SAS Variable Format                                                                                      | SAP ASE Data Type                                                     |
|----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| \$w., \$CHARw, \$VARYINGw.,<br>\$HEXw.                                                                   | VARCHAR(w)                                                            |
| DOLLARw.d                                                                                                | SMALLMONEY (where w < 6)<br>MONEY (where w >= 6)                      |
| datetime format                                                                                          | DATETIME                                                              |
| date format                                                                                              | DATE                                                                  |
| time format                                                                                              | TIME                                                                  |
| any numeric with a SAS format<br>name of <i>w.d</i> (where <i>d</i> > 0 and <i>w</i> > 10) or <i>w</i> . | NUMERIC(p,s)                                                          |
| any numeric with a SAS format<br>name of <i>w.d</i> (where <i>d</i> = 0 and <i>w</i> < 10)               | TINYINT (where w < 3)<br>SMALLINT (where w < 5)<br>INT (where w < 10) |
| any other numeric                                                                                        | FLOAT                                                                 |

You can override these default data types by using the [DBTYPE=](#) on page 536 data set option.

## ACCESS Procedure Data Conversions

This table shows the default SAS variable formats that SAS/ACCESS assigns to SAP ASE data types when you use the [ACCESS procedure](#).

**Table 34.6** PROC ACCESS: Default SAS Formats for SAP ASE Server Data Types

| SAP ASE Column Type | SAS Data Type | Default SAS Format                                           |
|---------------------|---------------|--------------------------------------------------------------|
| CHAR( <i>n</i> )    | character     | \$ <i>n</i> . ( <i>n</i> <= 200)<br>\$200. ( <i>n</i> > 200) |
| VARCHAR( <i>n</i> ) | character     | \$ <i>n</i> . ( <i>n</i> <= 200)<br>\$200. ( <i>n</i> > 200) |
| BIT                 | numeric       | 1.0                                                          |
| TINYINT             | numeric       | 4.0                                                          |
| SMALLINT            | numeric       | 6.0                                                          |
| INT                 | numeric       | 11.0                                                         |
| FLOAT               | numeric       | BEST22.                                                      |
| REAL                | numeric       | BEST11.                                                      |
| SMALLMONEY          | numeric       | DOLLAR12.2                                                   |
| MONEY               | numeric       | DOLLAR24.2                                                   |
| SMALLDATETIME       | numeric       | DATETIME21.2                                                 |
| DATETIME            | numeric       | DATETIME21.2                                                 |

The ACCESS procedure also supports SAP ASE user-defined data types. The ACCESS procedure uses the SAP ASE data type on which a user-defined data type is based in order to assign a default SAS format for columns.

The DECIMAL, NUMERIC, and TEXT data types are not supported in PROC ACCESS. The TIMESTAMP data type is not displayed in PROC ACCESS.

---

## DBLOAD Procedure Data Conversions

This table shows the default SAP ASE data types that SAS/ACCESS assigns to SAS variable formats when you use the [DBLOAD procedure on page 1455](#).

**Table 34.7** PROC DBLOAD: Default SAP ASE Data Types for SAS Variable Formats

| SAS Variable Format                           | SAP ASE Data Type |
|-----------------------------------------------|-------------------|
| \$w., \$CHARw., \$VARYINGw.,<br>\$HEXw.       | CHAR(w)           |
| w.                                            | TINYINT           |
| w.                                            | SMALLINT          |
| w.                                            | INT               |
| w.                                            | FLOAT             |
| w.d                                           | FLOAT             |
| IBw.d, PIBw.d                                 | INT               |
| FRACT, E format, and other<br>numeric formats | FLOAT             |
| DOLLARw.d, w<=12                              | SMALLMONEY        |
| DOLLARw.d, w>12                               | MONEY             |
| any datetime, date, or time<br>format         | DATETIME          |

The DBLOAD procedure also supports SAP ASE user-defined data types. Use the **TYPE=** statement to specify a user-defined data type.

---

## Data Returned as SAS Binary Data with Default Format \$HEX

- BINARY
- VARBINARY
- IMAGE

---

## Data Returned as SAS Character Data

- NCHAR
- NVARCHAR

---

## Inserting TEXT into SAP ASE from SAS

You can insert only TEXT data into an SAP ASE table by using the **BULKLOAD=** data set option, as in this example:

```
data yourlib.newtable(bulkload=yes);  
    set work.sasbigtext;  
run;
```

If you do not use the **BULKLOAD=** option, you receive this error message:

```
ERROR: Object not found in database. Error Code: -2782  
An untyped variable in the PREPARE statement 'S401bcf78'  
is being resolved to a TEXT or IMAGE type.  
This is illegal in a dynamic PREPARE statement.
```

---

## National Language Support for SAP ASE

To support output and update processing from SAS into SAP ASE in languages other than English, special setup steps are required. These setup steps are required so that date, time, and datetime values can be processed correctly. In SAS, you must ensure that the **DFLANG=** system option is set to the correct language. A system administrator can specify this globally administrator or a user can specify it within a single SAS session. In SAP ASE, the default client language, set in the *locales.dat* file, must match the language that is used in SAS.

# SAS/ACCESS Interface to SAP HANA

---

|                                                                       |      |
|-----------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to SAP HANA</i> ..... | 1224 |
| <i>Introduction to SAS/ACCESS Interface to SAP HANA</i> .....         | 1224 |
| <i>LIBNAME Statement for the SAP HANA Engine</i> .....                | 1225 |
| Overview .....                                                        | 1225 |
| Arguments .....                                                       | 1225 |
| SAP HANA LIBNAME Statement Examples .....                             | 1231 |
| <i>Data Set Options for SAP HANA</i> .....                            | 1231 |
| <i>SQL Pass-Through Facility Specifics for SAP HANA</i> .....         | 1234 |
| Key Information .....                                                 | 1234 |
| CONNECT Statement Examples .....                                      | 1234 |
| <i>Understanding SAP HANA Update and Delete Rules</i> .....           | 1235 |
| <i>Autopartitioning Scheme for SAP HANA</i> .....                     | 1236 |
| Overview .....                                                        | 1236 |
| Autopartitioning Restrictions .....                                   | 1236 |
| Nullable Columns .....                                                | 1236 |
| Using WHERE Clauses .....                                             | 1236 |
| Using DBSLICEPARM= .....                                              | 1237 |
| Using DBSLICE= .....                                                  | 1237 |
| <i>SAP HANA Schema Flexibility</i> .....                              | 1237 |
| <i>Temporary Table Support for SAP HANA</i> .....                     | 1238 |
| <i>Passing SAS Functions to SAP HANA</i> .....                        | 1238 |
| <i>Passing Joins to SAP HANA</i> .....                                | 1239 |
| <i>Bulk Loading for SAP HANA</i> .....                                | 1240 |
| Overview .....                                                        | 1240 |
| Data Set Options for Bulk Loading .....                               | 1240 |
| <i>Naming Conventions for SAP HANA</i> .....                          | 1241 |
| <i>Data Types for SAP HANA</i> .....                                  | 1242 |
| Overview .....                                                        | 1242 |

|                                           |      |
|-------------------------------------------|------|
| Data Types for SAP HANA .....             | 1242 |
| Working with Geospatial Data in SAS ..... | 1245 |
| LIBNAME Statement Data Conversions .....  | 1246 |

---

# System Requirements for SAS/ACCESS Interface to SAP HANA

You can find information about system requirements for SAS/ACCESS Interface to SAP HANA in the following locations:

- [System Requirements for SAS/ACCESS Interface to SAP HANA with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

# Introduction to SAS/ACCESS Interface to SAP HANA

For available SAS/ACCESS features, see [SAP HANA supported features on page 115](#). For more information about SAP HANA, see your SAP HANA documentation.

**Note:** The SAS/ACCESS Interface to SAP HANA was implemented for SAS 9.4.

**Note:** When you use analytic views in SQL statements, the measures must be aggregated. Beginning in SAS 9.4M3, the SAS/ACCESS engine generates a default statement with aggregated measures based on the metadata about the analytic view.

In SAS 9.4M3, PARMSTRING= and PARMDEFAULT= LIBNAME options and PARMSTRING= and PARMDEFAULT= data set options are available. The PARMSTRING= options specify a quoted string of variable name and value pairs or a placeholder string. The PARMDEFAULT= options specify whether the SAP HANA engine should use the defaults for variables and parameters as specified in the metadata in SAP HANA. For more information, see the following documentation:

- [“PARMSTRING= LIBNAME Statement Option” on page 275](#)
- [“PARMSTRING= Data Set Option” on page 580](#)
- [“PARMDEFAULT= LIBNAME Statement Option” on page 274](#)
- [“PARMDEFAULT= Data Set Option” on page 579](#)

Beginning in SAS Viya 3.3, SAS/ACCESS Interface to SAP HANA includes SAS Data Connector to SAP HANA. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- “Where to Specify Data Connector Options” in *SAS Cloud Analytic Services: User’s Guide*
- “SAP HANA Data Connector” in *SAS Cloud Analytic Services: User’s Guide*

---

# LIBNAME Statement for the SAP HANA Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to SAP HANA supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing SAP HANA.

**LIBNAME** *libref saphana <connection-options> <LIBNAME-options>*;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *saphana*

specifies the SAS/ACCESS engine name for the SAP HANA interface.

---

**Note:** Support for the

*saphana*

engine name was added in SAS 9.4M2.

---

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the SAP HANA server in many different ways. Specify only one of these methods for each connection because they are mutually exclusive.

- SERVER=, PORT=, USER=, PASSWORD=
- SERVER=, INSTANCE=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD= [This method uses a configured ODBC data source to store the connections options.]
- NOPROMPT=
- PROMPT=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**SERVER=<'>server-name<'>**  
**SERVER=<'>server-name:port<'>**  
**SERVER='server-name:port;failover-server-name1:port;failover-server-name2:port'**

specifies the server name or IP address of the SAP HANA server to which you want to connect. If the server name contains spaces or nonalphanumeric characters or if it is an IP address, you must enclose it in quotation marks. You can include the port when you specify a server. The port number is  $3<\text{instance-number}>15$  (for example, 30015 for the instance number 00). To support failover, you can specify a list of hostnames, separated by a semicolon. If a host is not available, the next host from the list is used.

Alias: HOST, SERVER, SERVERNODE

Example: SERVERNODE= 'saph1.mycompany.com'

**PORt=port**

specifies the port number that is used to connect to the specified SAP HANA server. The default is used if you do not specify a port or instance number or if you do not include the port number in the server specification. The port for the standard SQL communication for client access is  $3<\text{instance-number}>15$ .

Restriction: Do not specify a value for both the PORT= and INSTANCE= options. If you specify a value for both PORT= and INSTANCE=, then the following error appears in the log:

ERROR: Invalid combination of connection errors. You cannot use the PORT with the INSTANCE option.

Alias: PORT

Default: 30015

Example: server='saph1.mycompany.com' port=30215

**INSTANCE=instance-number**

specifies the instance number of the SAP HANA database engine. The port number is  $3<\text{instance-number}>15$  (for example, 30015 for the instance number 00). If you specify the port number explicitly in either PORT= or SERVER=, then INSTANCE= is ignored and a warning is written to the SAS log.

Restriction: Do not specify INSTANCE= if you also specify a port value for PORT= or SERVER=. If you specify a value for both PORT= and INSTANCE=, then the following error appears in the log:

ERROR: Invalid combination of connection errors. You cannot use the PORT with the INSTANCE option.

**Alias:** INSTANCE

**Example:** server='saph1.mycompany.com' instance=02

**USER=<'>SAP-HANA-user-name<'>**

specifies the SAP HANA user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**Alias:** USER

**Example:** USER=HANAUSER1

**PASSWORD=<'>SAP-HANA-password<'>**

specifies the password that is associated with your SAP HANA user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**Aliases:** PASS, PASSWORD, PW, PWD

**DSN=<'>SAP-HANA-data-source<'>**

specifies the configured SAP HANA ODBC data source to which you want to connect. Use this option if you have existing SAP HANA ODBC data sources that are configured on your client. This method requires additional setup. You can perform setup either through the ODBC Administrator control panel on Windows platforms or through the odbc.ini file or a similarly named configuration file on UNIX platforms. It is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

**Alias:** DSN

**Example:** DSN=HANADSN1

Here is an example for an odbc.ini entry on UNIX:

[MYSAPHANA]

SERVENODE=saph1.mycompany.com:30315

**NOPROMPT=<'>SAP-HANA-connection-options<'>**

specifies connection options for your database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned. No dialog box is displayed to help you complete the connection string. Connection options are appended to the connection string that is used to connect to the SAP HANA database. You can use this option to specify special options by adding key-value pairs to the connection string.

**PROMPT=<'>SAP-HANA-connection-information<'>**

specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately try to connect to the DBMS. A dialog box is displayed in SAS windowing environment instead that contains the values that you entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the database.

**Restriction:** This option is not available on UNIX platforms.

**DRIVER=<'>driver<'>**

specifies the ODBC driver to use to connect to your database.

**ENCRYPTION=YES | NO**

specifies how communication is encrypted.

**Alias:** ENCRYPT

**Default:** NO

**SSLCRYPTOPROVIDER=SAPCRYPTO | OPENSSL**

specifies the cryptographic library provider to use for SSL connectivity.

**Note:** Based on the SAP documentation, the OPENSSL value is deprecated and it is recommended that you migrate to CommonCryptoLib. For more information, see your SAP documentation.

**Alias:** SSLCRYPTOPROVIDER, SSLPROVIDER

**SSLKEYSTORE=<'>file<'>**

specifies the path to the keystore file.

**Alias:** SSLKEYSTORE

**ODBC driver default:** \$HOME/.ssl/key.pem

**SSLTRUSTSTORE=<'>file<'>**

specifies the path to the truststore file.

**Alias:** SSLTRUSTSTORE

**ODBC driver default:** \$HOME/.ssl/trust.pem

**SSLVALIDATECERTIFICATE=YES | NO**

indicates whether to validate the certificate of the communication partner.

**Alias:** SSLVALIDATECERTIFICATE

**ODBC driver default:** NO

**SSLHOSTNAMEINCERTIFICATE=<'>string<'>**

specifies the host-name certificate of the keystore.

**Alias:** SSLHOSTNAMEINCERT, SSLHOSTNAMEINCERTIFICATE

**SSLCREATESELFIGNEDCERTIFICATE=YES | NO**

specifies whether to create a self-signed certificate if the keystore cannot be found.

**Alias:** SSLCREATECERT, SSLCREATESELFIGNEDCERTIFICATE

#### *LIBNAME-options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to SAP HANA with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 35.1** SAS/ACCESS LIBNAME Options for SAP HANA

| Option  | Default Value | Valid in CONNECT |
|---------|---------------|------------------|
| ACCESS= | none          |                  |

| Option               | Default Value                                      | Valid in CONNECT |
|----------------------|----------------------------------------------------|------------------|
| AUTHDOMAIN=          | none                                               |                  |
| AUTOCOMMIT=          | NO                                                 | •                |
| CHAR_AS_NCHAR=       | NO                                                 |                  |
| CONNECTION=          | SHAREDREAD                                         | •                |
| CONNECTION_GROUP=    | none                                               | •                |
| CONNECTION_GROUP=    | none                                               |                  |
| CONOPTS=             | none                                               | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows) |                  |
| DBCONINIT=           | none                                               | •                |
| DBCONTERM=           | none                                               | •                |
| DBCREATE_TABLE_OPTS= | none                                               |                  |
| DBGEN_NAME=          | DBMS                                               | •                |
| DBINDEX=             | NO                                                 |                  |
| DBLIBINIT=           | none                                               |                  |
| DBLIBTERM=           | none                                               |                  |
| DBMAX_TEXT=          | 1024                                               | •                |
| DBMSTEMP=            | NO                                                 |                  |
| DBNULLKEYS=          | YES                                                |                  |
| DBPROMPT=            | NO                                                 | •                |
| DBNULLWHERE=         | YES                                                |                  |
| DBSASLABEL=          | COMPAT                                             |                  |
| DBSLICEPARM=         | THREADED_APPS,2 or THREADED_APPS,3                 |                  |
| DEFER=               | NO                                                 | •                |

| Option                    | Default Value                                                         | Valid in CONNECT |
|---------------------------|-----------------------------------------------------------------------|------------------|
| DELETE_MULT_ROWS=         | NO                                                                    |                  |
| DIRECT_EXE=               | none                                                                  |                  |
| DIRECT_SQL=               | YES                                                                   |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                                    |                  |
| INSERTBUFF=               | 1                                                                     |                  |
| LOGIN_TIMEOUT=            | 0                                                                     | •                |
| MULTI_DATASRC_OPT=        | NONE                                                                  |                  |
| PARMDEFAULT=              | none                                                                  |                  |
| PARMSTRING=               | none                                                                  |                  |
| PRESERVE_COL_NAMES=       | NO (see <a href="#">Naming Conventions for SAP HANA</a> on page 1241) |                  |
| PRESERVE_TAB_NAMES=       | NO (see <a href="#">Naming Conventions for SAP HANA</a> on page 1241) |                  |
| QUERY_TIMEOUT=            | 0                                                                     | •                |
| QUOTE_CHAR=               | none                                                                  |                  |
| READ_LOCK_TYPE=           | ROW                                                                   | •                |
| READBUFF=                 | 0                                                                     | •                |
| REREAD_EXPOSURE=          | NO                                                                    | •                |
| SCHEMA=                   | none                                                                  |                  |
| SPOOL=                    | YES                                                                   |                  |
| SQL_FUNCTIONS=            | none                                                                  |                  |
| SQL_FUNCTIONS_COPY=       | none                                                                  |                  |
| SQLGENERATION=            | none                                                                  |                  |
| STRINGDATES=              | NO                                                                    | •                |

| Option              | Default Value | Valid in CONNECT |
|---------------------|---------------|------------------|
| SUB_CHAR=           | none          |                  |
| TABLE_TYPE=         | none          |                  |
| TRACE=              | NO            | •                |
| TRACEFILE=          | none          | •                |
| UPDATE_LOCK_TYPE=   | ROW           | •                |
| UPDATE_MULT_ROWS=   | NO            |                  |
| USE_ODBC_CL=        | NO            |                  |
| UTILCONN_TRANSIENT= | NO            |                  |

## SAP HANA LIBNAME Statement Examples

In this example, SERVER=, DATABASE=, USER=, and PASSWORD= are connection options. No DSN style is specified. This is the default method, which is recommended.

```
libname A1 saphana server=mysrv1
      port=30015 user=myusr1 password='mypwd1';
```

This example requires that you specify a DSN style.

```
libname B1 saphana dsn=hnatest user=myusr1 password=mypwd1;
```

Here is an example of the LIBNAME statement using the PARMSTRING= option:

```
libname a saphana user=userid password=xxxx server=server-name
      instance=02 preserve_tab_names=yes parmstring="parm_price=30"
      preserve_col_names=yes;
```

## Data Set Options for SAP HANA

All SAS/ACCESS data set options in this table are supported for SAP HANA. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

| Option                       | Default Value                                                                   |
|------------------------------|---------------------------------------------------------------------------------|
| BL_CONTROL_FIELD_DELIMITER=  | , (comma)                                                                       |
| BL_CONTROL_RECORD_DELIMITER= | '\n' (newline)                                                                  |
| BL_CONTROL_QUOTATION_MARK=   | ' (single quotation mark)                                                       |
| BL_FILE_BADFILE=             | creates a file in the current directory or with the default file specifications |
| BL_FILE_CONTROLFILE=         | creates a file in the current directory or with the default file specifications |
| BL_FILE_DATAFILE=            | creates a file in the current directory or with the default file specifications |
| BL_FILE_DEFAULT_DIR=         | NO                                                                              |
| BL_FILE_DELETE_DATAFILE=     | '/tmp/'                                                                         |
| BL_IMPORT_BATCH_SIZE=        | none                                                                            |
| BL_IMPORT_OPTIONS=           | none                                                                            |
| BL_IMPORT_TABLE_LOCK=        | NO                                                                              |
| BL_IMPORT_TYPE_CHECK=        | NO                                                                              |
| BL_SFTP_HOST=                | none                                                                            |
| BL_SFTP_OPTIONS=             | none                                                                            |
| BL_SFTP_USER=                | none                                                                            |
| BL_SFTP_WAIT_MILLISECONDS=   | none                                                                            |
| BULKLOAD=                    | NO                                                                              |
| DBCOMMIT=                    | LIBNAME option value                                                            |
| DBCONDITION=                 | none                                                                            |
| DBCREATE_TABLE_OPTS=         | LIBNAME option value                                                            |
| DBFORCE=                     | NO                                                                              |
| DBGEN_NAME=                  | DBMS                                                                            |
| DBINDEX=                     | NO                                                                              |

| Option                    | Default Value                              |
|---------------------------|--------------------------------------------|
| DBKEY=                    | none                                       |
| DBLABEL=                  | NO                                         |
| DBLARGETABLE=             | none                                       |
| DBMAX_TEXT=               | 1024                                       |
| DBNULL=                   | YES                                        |
| DBNULLKEYS=               | LIBNAME option value                       |
| DBNULLWHERE=              | LIBNAME option value                       |
| DBPROMPT=                 | LIBNAME option value                       |
| DBSASLABEL=               | LIBNAME option value                       |
| DBSASTYPE=                | see “Data Types for SAP HANA” on page 1242 |
| DBSLICE=                  | none                                       |
| DBSLICEPARM=              | THREADED_APPS,2 or THREADED_APPS,3         |
| DBTYPE=                   | see “Data Types for SAP HANA” on page 1242 |
| ERRLIMIT=                 | 1                                          |
| IGNORE_READ_ONLY_COLUMNS= | NO                                         |
| INSERTBUFF=               | LIBNAME option value                       |
| NULLCHAR=                 | SAS                                        |
| NULLCHARVAL=              | a blank character                          |
| PARMDEFAULT=              | none                                       |
| PARMSTRING=               | none                                       |
| PRESERVE_COL_NAMES=       | LIBNAME option value                       |
| QUERY_TIMEOUT=            | LIBNAME option value                       |
| READ_LOCK_TYPE=           | LIBNAME option value                       |

| Option            | Default Value        |
|-------------------|----------------------|
| READBUFF=         | LIBNAME option value |
| SASDATEFMT=       | none                 |
| SCHEMA=           | LIBNAME option value |
| SUB_CHAR=         | none                 |
| TABLE_TYPE=       | none                 |
| UPDATE_LOCK_TYPE= | LIBNAME option value |

---

## SQL Pass-Through Facility Specifics for SAP HANA

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the SAP HANA interface.

- The *dbms-name* is **SAPHANA**.
- PROC SQL supports multiple connections to SAP HANA. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default SAPHANA alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).

---

## CONNECT Statement Examples

This example connects to SAP HANA and then disconnects from it.

```
proc sql noerrorstop;
  connect to saphana as x1(server=mysrv1
                           port=30015 user=mysur1 password='mypwd1');
  disconnect from x1;
  quit;
```

This next example connects to SAP HANA, executes some SQL statements, and then disconnects from SAP HANA.

```
proc sql noerrorstop;
connect to saphana as x1(server=mysrv1
                           port=30015 user=mysurl password='mypwd1'

                           execute (CREATE TABLE t1 (no int primary key, state varchar(10))) by
                           x1;
                           execute (INSERT INTO t1 values (1, 'USA')) by x1;
                           execute (INSERT INTO t1 values (2, 'CHN')) by x1;
                           select * from connection to x1 (SELECT * FROM t1 ORDER BY no);

                           disconnect from x1;
                           quit;
```

## Understanding SAP HANA Update and Delete Rules

To avoid data integrity problems when updating or deleting data, you need to specify a primary key on your table.

This example uses DBTYPE= to create the primary key.

```
libname invty saphana server=mysrv1 port=30015 user=mysurl;

proc sql;
drop table invty.STOCK23;
quit;

data invty.STOCK23 (DBTYPE=(RECDATE="date not null,primary key(RECDATE)"));
  input PARTNO $ DESCX $ INSTOCK @17
        RECDATE date7. @25 PRICE;
  format RECDATE date7.;
  datalines;

K89R  seal      34  27jul95  245.00
M447  sander    98  20jun95  45.88
LK43   filter    121 19may96  10.99
MN21   brace     43  10aug96  27.87
BC85   clamp     80  16aug96   9.55
KJ66   cutter     6  20mar96  24.50
UYN7   rod       211 18jun96  19.77
JD03   switch    383 09jan97  13.99
BV1I   timer     26  03jan97  34.50
;
```

These next examples show how you can update the table now that STOCK23 has a primary key.

```
proc sql;
update invty.STOCK23 set price=price*1.1 where INSTOCK > 50;
```

```

quit;

proc sql;
delete from invty.STOCK23 where INSTOCK > 150;
quit;

```

# Autopartitioning Scheme for SAP HANA

## Overview

Autopartitioning for SAS/ACCESS Interface to SAP HANA is a modulo (MOD) function method. For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

## Autopartitioning Restrictions

SAS/ACCESS Interface to SAP HANA places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- INTEGER, SMALLINT, and TINYINT columns are given preference.
- You can use DECIMAL, DOUBLE, FLOAT, NUMERIC, and REAL columns for partitioning. These types are converted to BIGINT by using the CAST function.

## Nullable Columns

If you select a nullable column for autopartitioning, the OR<column-name>IS NULL SQL statement is appended at the end of the SQL code that is generated for the threaded Read. This ensures that any possible NULL values are returned in the result set.

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in the WHERE clause. For example, the following DATA step could not use a threaded Read to retrieve the data because all numeric columns in the table are in the WHERE clause.

```
data work.locemp;
```

```

set hanalib.MYEMPS;
where EMPNUM<=30 and ISTENURE=0
      and SALARY<=35000 and NUMCLASS>2;
run;

```

## Using DBSLICEPARM=

SAS/ACCESS Interface to SAP HANA defaults to three threads when you use autopartitioning, but do not specify a maximum number of threads in **DBSLICEPARM=** to use for the threaded Read.

## Using DBSLICE=

You might achieve the best possible performance when using threaded Reads by specifying the **DBSLICE=** option for SAP HANA in your SAS operation. This is especially true if your table uses multiple database partitions. You can also use the **DBSLICE=** clause to specify the WHERE clause for each partition, as shown in this example.

```

proc print data=trilib.MYEMPS(DBSLICE=( "EMPNUM BETWEEN 1 AND 33"
   "EMPNUM BETWEEN 34 AND 66" "EMPNUM BETWEEN 67 AND 100")) ;
run;

```

The methods and examples described in **DBSLICE=** work well in cases where the table that you want to read is not stored in multiple partitions in your DBMS. These methods also give you flexibility in column selection. For example, if you know that the STATE column in your employee table contains only a few distinct values, you can customize your **DBSLICE=** clause accordingly.

```

datawork.locemp;
set hanalib2.MYEMP(DBSLICE=( "STATE='FL'" "STATE='GA'"
                           "STATE='SC'" "STATE='VA'" "STATE='NC'"));
where EMPNUM<=30 and ISTENURE=0 and SALARY<=35000 and NUMCLASS>2;
run;

```

# SAP HANA Schema Flexibility

The SAS/ACCESS engine for SAP HANA limits the number of columns in a table or a view to 64,000. Regular SAP HANA tables and views have a limit of 1,000 columns. If you try to create tables with more columns, the SAP HANA ODBC driver returns an error.

SAP HANA tables with schema flexibility can have up to 64,000 columns. To create SAP HANA tables with schema flexibility, use the following LIBNAME option or data set option:

```
DBCREATE_TABLE_OPTS='WITH SCHEMA FLEXIBILITY'
```

# Temporary Table Support for SAP HANA

SAS/ACCESS Interface to SAP HANA supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## Passing SAS Functions to SAP HANA

SAS/ACCESS Interface to SAP HANA passes the following SAS functions to SAP HANA for processing. Where the SAP HANA function name differs from the SAS function name, the SAP HANA name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                                          |                                             |
|------------------------------------------|---------------------------------------------|
| ** (POWER(base, exponent))               | LOG2 (LOG(2, value))                        |
| ABS                                      | LOWCASE (LOWER)                             |
| ARCOS (ACOS)                             | MAX                                         |
| ARSIN (ASIN)                             | MIN                                         |
| ATAN                                     | MINUTE                                      |
| ATAN2                                    | MOD (see note)                              |
| AVG                                      | MONTH                                       |
| BYTE (CHAR(CAST(expression AS INTEGER))) | QTR (CAST(TO_CHAR(value, 'Q') AS SMALLINT)) |
| CEIL                                     | SECOND                                      |
| COALESCE                                 | SIGN                                        |
| COS                                      | SIN                                         |
| COSH                                     | SINH                                        |
| COT                                      | SQRT                                        |
| COUNT                                    | STD (STDDEV)                                |
| DAY (DAYOFMONTH)                         | STRIP (TRIM)                                |
| EXP                                      | SUBSTR (SUBSTRING)                          |
| FLOOR                                    | SUM                                         |
| HOUR                                     | TAN                                         |
| INDEX (LOCATE)                           | TANH                                        |
| LEFT (RPAD(LTRIM(value), LENGTH(value))) | TRANWRD (REPLACE)                           |
| LENGTH                                   | TRIMN (RTRIM)                               |
| LENGTHC (LENGTH)                         | UPCASE (UPPER)                              |
| LENGTHN (LENGTH)                         | VAR                                         |
| LOG (LN)                                 | WEEKDAY (DAYOFWEEK)                         |
| LOG10 (LOG(10, value))                   | YEAR                                        |

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to SAP HANA. Due to incompatibility in date and time functions between SAP HANA and SAS, SAP HANA might not process them correctly. Check your results to determine whether these functions are working as expected.

|                                     |                                     |
|-------------------------------------|-------------------------------------|
| COMPRESS (REPLACE)                  | TIME (CURRENT_TIME)                 |
| DATE (CURRENT_DATE)                 | TIMEPART (CAST(expression AS TIME)) |
| DATEPART (CAST(expression AS DATE)) | TODAY (CURRENT_DATE)                |
| DATETIME (CURRENT_TIMESTAMP)        | TRANSLATE                           |
| SYSDATE                             |                                     |

## Passing Joins to SAP HANA

For a multiple libref join to pass to SAP HANA, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- instance (INSTANCE=)
- port (PORT=)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

# Bulk Loading for SAP HANA

---

## Overview

Bulk loading is the fastest way to insert large numbers of rows into an SAP HANA table. To use the bulk-load facility, specify **BULKLOAD=YES**. The bulk-load facility uses the SAP HANA IMPORT statement and Secure Shell (SSH) File Transfer Protocol (FTP), or SFTP, to move data from the client to the SAP HANA database.

---

## Data Set Options for Bulk Loading

Here are the SAP HANA bulk-load data set options. For detailed information about these options, see [Data Set Options for Relational Databases on page 370](#).

- [BL\\_FILE\\_BADFILE=](#)
- [BL\\_CONTROL\\_FIELD\\_DELIMITER=](#)
- [BL\\_CONTROL\\_RECORD\\_DELIMITER=](#)
- [BL\\_CONTROL\\_QUOTATION\\_MARK=](#)
- [BL\\_FILE\\_CONTROLFILE=](#)
- [BL\\_FILE\\_DATAFILE=](#)
- [BL\\_FILE\\_DEFAULT\\_DIR=](#)
- [BL\\_FILE\\_DELETE\\_DATAFILE=](#)
- [BL\\_IMPORT\\_BATCH\\_SIZE=](#)
- [BL\\_IMPORT\\_OPTIONS=](#)
- [BL\\_IMPORT\\_TABLE\\_LOCK=](#)
- [BL\\_IMPORT\\_TYPE\\_CHECK=](#)
- [BL\\_SFTP\\_HOST=](#)
- [BL\\_SFTP\\_OPTIONS=](#)
- [BL\\_SFTP\\_USER=](#)
- [BL\\_SFTP\\_WAIT\\_MILLISECONDS=](#)
- [BULKLOAD=](#)

# Naming Conventions for SAP HANA

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The SAP HANA interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options determine how SAS/ACCESS Interface to SAP HANA handles case sensitivity. Although SAP HANA is not case-sensitive, all names are stored in uppercase if you do not enclose them in double quotation marks. These options also control whether you can use an SAP HANA reserved word as an identifier. For more information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).

SAP HANA objects include tables, views, synonyms, columns, indexes, functions, procedures, users, roles, and more. To represent names for these objects in the SQL statement, SAP HANA uses delimited identifiers and undelimited identifiers. An undelimited identifier represents an undelimited table or column name.

Follow these naming conventions:

- An identifier must be from 1 to 127 characters long.
- An undelimited name must begin with a letter (A through Z). It cannot contain any symbols except digits or an underscore (\_).
- A delimited name must be enclosed within the double quotation ("") delimiter. It can contain any character, including special characters (for example, "AB\$%CD").
- Names are not case sensitive. For example, CUSTOMER and Customer are the same, but object names are converted to uppercase when they are stored in the SAP HANA database. However, if you enclose a name in quotation marks, it is case sensitive.
- A name cannot be an SAP HANA reserved word.

**Note:** If PRESERVE\_COL\_NAMES= is enabled (set to YES), then a column name can be a reserved word. Similarly, if PRESERVE\_TAB\_NAMES= is enabled (set to YES), then a table name can be a reserved word. By default, these options are disabled.

- A name cannot be the same as another SAP HANA object that has the same type.

For more information, see your SAP HANA documentation.

---

# Data Types for SAP HANA

---

## Overview

Every column in a table has a name and a data type. The data type tells SAP HANA how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about SAP HANA data types, null and default values, and data conversions.

---

# Data Types for SAP HANA

---

## Overview

Here are the data types that the SAP HANA engine supports.

- Character data:

VARCHAR(n)  
NCHAR  
NVARCHAR(n)  
ALPHANUM  
SHORTTEXT

**Note:** The SAS/ACCESS for SAP HANA engine calculates the length of SAP HANA NCHAR and NVARCHAR data types by multiplying the number of SAP HANA characters by 3 when reading data to a SAS session with UTF8. This calculation provides enough space to read an SAP character-based NCHAR or NVARCHAR type into a SAS byte-based character variable. For example, an SAP HANA character variable of 3 characters would be read into a SAS byte-based character variable calculated as length 9. When writing SAS data to SAP HANA, the engine needs to specify an NCHAR or NVARCHAR variable length that allows all of the SAS byte-based character column to the SAS HANA character column. For example, a SAS character variable that holds 9 characters might contain 9 single-byte characters. The engine ensures enough space in the SAP HANA variable length by writing the data using a length of 9 characters. To override the default type determined by the SAS/ACCESS engine for SAP HANA, use the “[DBTYPE= Data Set Option](#)” on page 536 .

- Binary data: VARBINARY
- Large object (LOB) data:

BLOB (binary large object)  
 CLOB (character large object)  
 NCLOB  
 TEXT

■ Numeric data:

|                                                        |                   |
|--------------------------------------------------------|-------------------|
| TINYINT                                                | SMALLDECIMAL      |
| SMALLINT                                               | REAL              |
| INTEGER                                                | DOUBLE            |
| BIGINT                                                 | FLOAT( <i>n</i> ) |
| DECIMAL( <i>precision,scale</i> ) or DEC( <i>p,s</i> ) |                   |

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

---

■ Date, time, and timestamp data:

DATE  
 TIME  
 SECONDDATE  
 TIMESTAMP

## SAP HANA Null Values

Many relational database management systems have a special value called NULL. A DBMS NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a DBMS NULL value, it interprets it as a SAS missing value.

In most relational databases, columns can be specified as NOT NULL so that they require data (they cannot contain NULL values). When a column is specified as NOT NULL, the DBMS does not add a row to the table unless the row has a value for that column. When creating a DBMS table with SAS/ACCESS, you can use the **DBNULL=** data set option to indicate whether NULL is a valid value for specified columns.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how the DBMS handles SAS missing character values, use the **NULCHAR=** and **NULCHARVAL=** data set options.

## Geospatial Data

SAS/ACCESS Interface to SAP HANA can import and export geospatial data of the type ST\_GEOMETRY. For example, you can export the following subtypes of geospatial data to SAP HANA tables:

### ST\_POINT

represents a single location (point) that is represented by a pair of latitude and longitude coordinates. Data of the geospatial type point in an SAP HANA table might look like the following example:

```
SHAPE.ST_ASJSON()
{"type": "Point", "coordinates": [84.24823999999996,18.79219399999999]}
```

In a SAS table, the latitude and longitude coordinates are each stored in a numeric variable.

### ST\_POLYGON

represents a set of points that specify the perimeter of an area. The points specify the vertices of a closed polygon. A record for an item of type ST\_POLYGON is made of multiple pairs of coordinates that specify the points (vertices) in the polygon. The first and last points must contain the same values. Geospatial data of type polygon in an SAP HANA table might look like the following example:

```
SHAPE.ST_ASWKT()
POLYGON ((  
84.24824 18.79219, 84.25500 18.78855, 84.25984 18.78489, 84.28588 18.78654,  
84.29696 18.79453, 84.31144 18.79190, 84.32205 18.79154, 84.33023 18.80136,  
84.33841 18.80701, 84.34510 18.82981, 84.35423 18.84336, 84.35228 18.85309,  
84.34839 18.86280, 84.34835 18.87626, 84.35803 18.86846,  
...  
83.93397 18.64837, 83.93369 18.68782, 83.94396 18.69438, 83.96128 18.67464,  
83.99588 18.64173, 84.01654 18.63514, 84.04403 18.63511, 84.05769 18.64825,  
84.10570 18.66136, 84.11241 18.68436, 84.11224 18.70737, 84.11905 18.71723,  
84.18427 18.72703, 84.19458 18.72702, 84.19444 18.74674, 84.22868 18.76315,  
84.24824 18.79219 ))
```

This data would be read from multiple records in a SAS data set into a single record in the SAP HANA table. Each row in the SAS data set contains the coordinates from one point (vertex) of a polygon. Multiple rows from the SAS data set are combined and exported into SAP HANA into a variable of data type ST\_POLYGON.

Other SAP HANA ST\_GEOMETRY subtypes, in addition to the examples shown here, can be supported.

## Working with Geospatial Data in SAS

### Overview of Importing Geospatial Data into SAS

In SAP HANA, geospatial data (data of type ST\_GEOMETRY) can take a number of subtypes, including point (ST\_POINT), polygon (ST\_POLYGON), and multi-polygon (ST\_MULTIPOLYGON). Geospatial data consists of pairs of latitude and longitude coordinate values. These coordinates can be read into numeric variables, such as X and Y, that respectively store the latitude and longitude values for each point.

Typically, you must convert relevant parts of geospatial data input into numeric or character data in SAS. If multiple geospatial points (representing longitude and latitude coordinates) are imported, such as for the subtype ST\_POLYGON, the best practice is to import each into separate rows. You can combine the rows before exporting the data back to SAP HANA.

To import geospatial data, import the values by using PROC SQL. Alternatively, you can import SHP files by using PROC MAPIMPORT. For more information, see [SAS SQL Procedure User's Guide](#), [SAS/GRAFH: Reference](#), and your SAP HANA documentation.

### Example 1: Export Point Data into SAP HANA

The following code shows how to export point coordinates into a table in SAP HANA:

```
execute (insert into "MY_GEOSPATIAL_FROM_SAS"(segment, area, flag,
  party, pc_code, pc_hname, pc_name, pc_no, pc_type, st_code,
  st_name , shape)
        (select segment, area, flag, party, pc_code, pc_hname, pc_name, pc_no,
               pc_type, st_code, st_name,  new ST_POINT('POINT('||x||' '||y||')
        from #temp_a))
by saphana;
```

In this example, the numeric variables X and Y in the SAS data set specify the latitude and longitude coordinates for a point. The data from table SAS data set TEMP\_A is exported into an SAP HANA table MY\_GEOSPATIAL\_FROM\_SAS that contains geospatial data of type ST\_POINT.

## Example 2: Export Polygon Data into SAP HANA

In SAS, data for a polygon that represents a city is stored in multiple rows. Each row contains the latitude and longitude coordinates for a point (vertex) of the polygon that specifies the city's perimeter. All of the rows for one city must be combined into a single row in the target SAP HANA table. Add the coordinates to a single instance of a variable of type ST\_POLYGON.

Before exporting coordinates for geospatial data of type ST\_POLYGON, generate an intermediate SAS data set that combines the coordinate data from each row into a single record for each polygon. You can do this using FIRST. and LAST. processing and concatenation functions.

In this example, the coordinates for each vertex of the polygon were originally stored in separate variables for the latitude and longitude. These pairs of values are combined into a variable, ST\_POLY, that lists the pairs of coordinates separated by commas. A row in the intermediate data set might contain data that looks similar to this:

```
ST_POLY = 84.24823999999996 18.79219000000002, 84.22867999999997
          18.76315000000000, 84.19444000000000 18.74673999999999
City = Cary
Zip = 27513
State = NC
```

The following code shows how to export polygon data from the intermediate SAS data set into a table in SAP HANA:

```
execute (insert into "MY_GEOSP_POLYGON_FROM_SAS" (segment, area,
  flag, party, pc_code, pc_hname, pc_name, pc_no, pc_type,
  st_code, st_name , shape)
        (select segment, area, flag, party, pc_code, pc_hname, pc_name, pc_no,
               pc_type, st_code, st_name ,
               new ST_POLYGON('POLYGON(|||to_char(ST_POLY)|||)')
               from #temp_a)
        by saphana;
```

In this example, the data in the ST\_POLY variable in the intermediate SAS data set TEMP\_A is exported into an SAP HANA table called MY\_GEOSP\_POLYGON\_FROM\_SAS. This table contains geospatial data of type ST\_POLYGON.

---

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to SAP HANA assigns to SAS variables when using the [LIBNAME statement](#) to read from an SAP HANA table. These default formats are based on SAP HANA column attributes.

**Table 35.2 LIBNAME Statement: Default SAS Formats for SAP HANA Data Types**

| SAP HANA Data Type                                                     | SAS Data Type | Default SAS Format                                                                             |
|------------------------------------------------------------------------|---------------|------------------------------------------------------------------------------------------------|
| VARCHAR( <i>n</i> ) <sup>1</sup>                                       | character     | \$ <i>w</i> .                                                                                  |
| NVARCHAR( <i>n</i> ) <sup>1</sup>                                      | character     | \$ <i>w</i> .                                                                                  |
| ALPHANUM( <i>n</i> ) <sup>1</sup>                                      | character     | \$ <i>w</i> .                                                                                  |
| SHORTTEXT( <i>n</i> ) <sup>1</sup>                                     | character     | \$ <i>w</i> .                                                                                  |
| VARBINARY( <i>n</i> ) <sup>1</sup>                                     | character     | \$ <i>w</i> . (where <i>w</i> is $2^*n$ .)                                                     |
| BLOB                                                                   | character     | \$ <i>w</i> . (where <i>w</i> is the minimum of 32767 and the value of the DBMAX_TEXT= option) |
| CLOB <sup>3</sup>                                                      | character     | \$ <i>w</i> . (where <i>w</i> is the minimum of 32767 and the value of the DBMAX_TEXT= option) |
| NCLOB                                                                  | character     | \$ <i>w</i> . (where <i>w</i> is the minimum of 32767 and the value of the DBMAX_TEXT= option) |
| TEXT                                                                   | character     | \$ <i>w</i> . (where <i>w</i> is the minimum of 32767 and the value of the DBMAX_TEXT= option) |
| TINYINT                                                                | numeric       | 4.                                                                                             |
| SMALLINT                                                               | numeric       | 6.                                                                                             |
| INTEGER                                                                | numeric       | 11.                                                                                            |
| BIGINT                                                                 | numeric       | 20.                                                                                            |
| DECIMAL( <i>precision,scale</i> )<br>or DEC( <i>p,s</i> ) <sup>2</sup> | numeric       | <i>w.d</i>                                                                                     |
| SMALLDECIMAL                                                           | numeric       | <i>w.d</i>                                                                                     |
| REAL                                                                   | numeric       | none                                                                                           |
| DOUBLE                                                                 | numeric       | none                                                                                           |
| FLOAT( <i>n</i> )                                                      | numeric       | none                                                                                           |

| SAP HANA Data Type                                    | SAS Data Type | Default SAS Format |
|-------------------------------------------------------|---------------|--------------------|
| DATE                                                  | numeric       | DATE9.             |
| TIME                                                  | numeric       | TIME8.             |
| SECONDDATE                                            | numeric       | DATETIME20.        |
| TIMESTAMP                                             | numeric       | DATETIME26.7       |
| ST_Geometry (such as ST_POINT, ST_POLYGON, and so on) | numeric       | none               |

- 1 *n* in SAP HANA character data types is equivalent to *w* in SAS formats.
- 2 *p* and *s* in SAP HANA numeric data types are equivalent to *w* and *d* in SAS formats.
- 3 The value of the DBMAX\_TEXT= option can override these values.

# SAS/ACCESS Interface to SAP IQ

---

|                                                               |      |
|---------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to SAP IQ</i> | 1250 |
| <i>Introduction to SAS/ACCESS Interface to SAP IQ</i>         | 1250 |
| <b>LIBNAME Statement for the SAP IQ Engine</b>                | 1251 |
| Overview                                                      | 1251 |
| Arguments                                                     | 1251 |
| LIBNAME Statement Examples for SAP IQ                         | 1255 |
| <b>Data Set Options for SAP IQ</b>                            | 1256 |
| <b>SQL Pass-Through Facility Specifics for SAP IQ</b>         | 1258 |
| Key Information                                               | 1258 |
| CONNECT Statement Example                                     | 1258 |
| Requirement for ORDER BY Variables                            | 1259 |
| Special Catalog Queries                                       | 1259 |
| <b>Autopartitioning Scheme for SAP IQ</b>                     | 1260 |
| Overview                                                      | 1260 |
| Autopartitioning Restrictions                                 | 1260 |
| Nullable Columns                                              | 1260 |
| Using WHERE Clauses                                           | 1261 |
| Using DBSLICEPARM=                                            | 1261 |
| Using DBSLICE=                                                | 1261 |
| <b>Temporary Table Support for SAP IQ</b>                     | 1262 |
| <b>Passing SAS Functions to SAP IQ</b>                        | 1262 |
| <b>Passing Joins to SAP IQ</b>                                | 1263 |
| <b>Bulk Loading for SAP IQ</b>                                | 1263 |
| Loading                                                       | 1263 |
| Data Set Options for Bulk Loading                             | 1264 |
| Bulk Loading Examples                                         | 1264 |
| <b>Locking in the SAP IQ Interface</b>                        | 1265 |
| <b>Naming Conventions for SAP IQ</b>                          | 1266 |

|                                          |             |
|------------------------------------------|-------------|
| <b>Data Types for SAP IQ .....</b>       | <b>1267</b> |
| Overview .....                           | 1267        |
| Supported SAP IQ Data Types .....        | 1267        |
| SAP IQ Null Values .....                 | 1268        |
| LIBNAME Statement Data Conversions ..... | 1268        |

---

# System Requirements for SAS/ACCESS Interface to SAP IQ

You can find information about system requirements for SAS/ACCESS Interface to SAP IQ in the following locations:

- [System Requirements for SAS/ACCESS Interface to SAP IQ with SAS 9.4](#)
- [Third-Party Software Requirements for Use with SAS 9.4](#)

---

# Introduction to SAS/ACCESS Interface to SAP IQ

For available SAS/ACCESS features, see [SAP IQ supported features on page 116](#). For more information about SAP IQ, see your SAP IQ documentation.

---

**Note:** In [SAS 9.4M4](#), the SAS/ACCESS product name changed from SAS/ACCESS Interface to Sybase IQ to SAS/ACCESS Interface to SAP IQ to reflect the name change of the SAP IQ product.

---



---

**Note:** SAS/ACCESS Interface to SAP IQ is not included with SAS Viya.

---

# LIBNAME Statement for the SAP IQ Engine

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to SAP IQ supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing SAP IQ.

**LIBNAME** *libref sapiq* <connection-options> <LIBNAME-options>;

---

**Note:** The previous engine name, sybaseiq, is also accepted for this SAS/ACCESS engine.

---

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *sapiq*

specifies the SAS/ACCESS engine name for the SAP IQ interface.

### *connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the SAP IQ database in two ways. Specify only one of these methods for each connection because they are mutually exclusive.

- HOST=, SERVER=, DATABASE=, PORT=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

**HOST=<'>server-name<'>**

specifies the host name or IP address where the SAP IQ database is running. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: HOSTNAME=

**SERVER=<'>server-name<'>**

specifies the SAP IQ server name, also known as the engine name. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: ENGINENAME=

**DATABASE=<'>database-name<'>**

specifies the SAP IQ database that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORt=port**

specifies the port number that is used to connect to the specified SAP IQ database. If you do not specify a port, the default is 2638.

Aliases: SERVICE=, SERVICE\_NAME=

**USER=<'>SAP-IQ-user-name<'>**

specifies the SAP IQ user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: UID=

**PASSWORD=<'>SAP-IQ-password<'>**

specifies the password that is associated with your SAP IQ user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Aliases: PASS=, PWD=, PW=

**DSN=<'>SAP-IQ-data-source<'>**

specifies the configured SAP IQ ODBC data source to which you want to connect. Use this option if you have existing SAP IQ ODBC data sources that are configured on your client. This method requires additional setup—either through the ODBC Administrator control panel on Windows platforms or through the odbc.ini file or a similarly named configuration file on UNIX platforms. So it is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

Aliases: DATASRC=, DS=

**LIBNAME-options**

define how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to SAP IQ, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 36.1** SAS/ACCESS LIBNAME Options for SAP IQ

| Option               | Default Value                                      | Valid in CONNECT |
|----------------------|----------------------------------------------------|------------------|
| ACCESS=              | none                                               |                  |
| AUTHDOMAIN=          | none                                               |                  |
| AUTOCOMMIT=          | operation-specific                                 | •                |
| CONNECTION=          | SHAREDREAD                                         | •                |
| CONNECTION_GROUP=    | none                                               | •                |
| CONOPTS=             | none                                               | •                |
| DBCLIENT_MAX_BYTES=  | 1                                                  | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows) |                  |
| DBCONINIT=           | none                                               | •                |
| DBCONTERM=           | none                                               | •                |
| DBCREATE_TABLE_OPTS= | none                                               |                  |
| DBGEN_NAME=          | DBMS                                               | •                |
| DBINDEX=             | NO                                                 |                  |
| DBLIBINIT=           | none                                               |                  |
| DBLIBTERM=           | none                                               |                  |
| DBMAX_TEXT=          | 1024                                               | •                |
| DBMSTEMP=            | NO                                                 |                  |
| DBNULLKEYS=          | YES                                                |                  |
| DBPROMPT=            | NO                                                 | •                |
| DBSASLABEL=          | COMPAT                                             |                  |
| DBSLICEPARM=         | THREADED_APPS,2<br>or<br>THREADED_APPS,3           |                  |

| Option                    | Default Value                                | Valid in CONNECT |
|---------------------------|----------------------------------------------|------------------|
| DEFER=                    | NO                                           | •                |
| DELETE_MULT_ROWS=         | NO                                           |                  |
| DIRECT_EXE=               | none                                         |                  |
| DIRECT_SQL=               | YES                                          |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                                           |                  |
| INSERTBUFF=               | automatically calculated based on row length |                  |
| LOGIN_TIMEOUT=            | 0                                            | •                |
| MULTI_DATASRC_OPT=        | NONE                                         |                  |
| POST_STMT_OPTS=           | none                                         |                  |
| PRESERVE_COL_NAMES=       | see “Naming Conventions for SAP IQ”          |                  |
| PRESERVE_TAB_NAMES=       | see “Naming Conventions for SAP IQ”          |                  |
| QUERY_TIMEOUT=            | 0                                            | •                |
| QUOTE_CHAR=               | none                                         |                  |
| READ_ISOLATION_LEVEL=     | RC (see “Locking in the SAP IQ Interface”)   | •                |
| READ_LOCK_TYPE=           | ROW                                          |                  |
| READBUFF=                 | automatically calculated based on row length | •                |
| REREAD_EXPOSURE=          | NO                                           | •                |
| SCHEMA=                   | none                                         |                  |
| SPOOL=                    | YES                                          |                  |

| Option                  | Default Value                              | Valid in CONNECT |
|-------------------------|--------------------------------------------|------------------|
| SQL_FUNCTIONS=          | none                                       |                  |
| SQL_FUNCTIONS_COPY=     | none                                       |                  |
| STRINGDATES=            | NO                                         | •                |
| TRACE=                  | NO                                         | •                |
| TRACEFILE=              | none                                       | •                |
| UPDATE_ISOLATION_LEVEL= | RC (see “Locking in the SAP IQ Interface”) | •                |
| UPDATE_LOCK_TYPE=       | ROW                                        | •                |
| UPDATE_MULT_ROWS=       | NO                                         |                  |
| UTILCONN_TRANSIENT=     | NO                                         |                  |

## LIBNAME Statement Examples for SAP IQ

In this example, HOST=, SERVER=, DATABASE=, USER=, and PASSWORD= are connection options.

```
libname mydblib sapiq host=iqsvr1 server=iqsvr1_users
      db=users user=iqusrl password=iqpwd1;

proc print data=mydblib.customers;
  where state='CA';
run;
```

In the next example, DSN=, USER=, and PASSWORD= are connection options. The SAP IQ SQL data source is configured in the ODBC Administrator Control Panel on Windows platforms or in the odbc.ini file or a similarly named configuration file on UNIX platforms.

```
libname mydblib sapiq DSN=SAPIQSQL user=iqusrl password=iqpwd1;

proc print data=mydblib.customers;
  where state='CA';
run;
```

# Data Set Options for SAP IQ

All SAS/ACCESS data set options in this table are supported for SAP IQ. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 36.2** SAS/ACCESS Data Set Options for SAP IQ

| Option               | Default Value                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------|
| BL_CLIENT_DATAFILE=  | none                                                                                                            |
| BL_COLUMN_DELIMITER= | (pipe symbol)<br>Alias is BL_DELIMTER=.                                                                         |
| BL_DATAFILE=         | When BL_USE_PIPE=NO, creates a file in the current directory or with the default file specifications.           |
| BL_DELETE_DATAFILE=  | YES (only when BL_USE_PIPE=NO)                                                                                  |
| BL_OPTIONS=          | none                                                                                                            |
| BL_ROW_DELIMITER=    | \x0A\x02\x0A (line feed, Control-B, line feed)                                                                  |
| BL_SERVER_DATAFILE=  | creates a data file in the current directory or with the default file specifications (same as for BL_DATAFILE=) |
| BL_USE_PIPE=         | YES                                                                                                             |
| BULKLOAD=            | NO                                                                                                              |
| DBCOMMIT=            | LIBNAME option value                                                                                            |
| DBCONDITION=         | none                                                                                                            |
| DBCREATE_TABLE_OPTS= | LIBNAME option value                                                                                            |
| DBFORCE=             | NO                                                                                                              |
| DBGEN_NAME=          | DBMS                                                                                                            |
| DBINDEX=             | LIBNAME option value                                                                                            |

| <b>Option</b>             | <b>Default Value</b>                  |
|---------------------------|---------------------------------------|
| DBKEY=                    | none                                  |
| DBLABEL=                  | NO                                    |
| DBLARGETABLE=             | none                                  |
| DBMAX_TEXT=               | 1024                                  |
| DBNULL=                   | YES                                   |
| DBNULLKEYS=               | LIBNAME option value                  |
| DBPROMPT=<                | LIBNAME option value                  |
| DBSASTYPE=                | see "Data Types for SAP IQ"           |
| DBSLICE=                  | none                                  |
| DBSLICEPARM=              | THREADED_APPS,2 or<br>THREADED_APPS,3 |
| DBTYPE=                   | see "Data Types for SAP IQ"           |
| ERRLIMIT=                 | 1                                     |
| IGNORE_READ_ONLY_COLUMNS= | NO                                    |
| INSERTBUFF=               | LIBNAME option value                  |
| NULLCHAR=                 | SAS                                   |
| NULLCHARVAL=              | a blank character                     |
| POST_STMT_OPTS=           | none                                  |
| POST_TABLE_OPTS=          | none                                  |
| PRE_STMT_OPTS=            | none                                  |
| PRE_TABLE_OPTS=           | none                                  |
| PRESERVE_COL_NAMES=       | LIBNAME option value                  |
| QUERY_TIMEOUT=            | LIBNAME option value                  |
| READ_ISOLATION_LEVEL=     | LIBNAME option value                  |
| READ_LOCK_TYPE=           | LIBNAME option value                  |

| Option      | Default Value        |
|-------------|----------------------|
| READBUFF=   | LIBNAME option value |
| SASDATEFMT= | none                 |
| SCHEMA=     | LIBNAME option value |

---

## SQL Pass-Through Facility Specifics for SAP IQ

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the SAP IQ interface.

- The *dbms-name* is **SAPIQ**.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to SAP IQ. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default `sapiq` alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection-options](#).

---

### CONNECT Statement Example

This example uses the DBCON alias to connection to the `iqsvr1` SAP IQ database and execute a query. The connection alias is optional.

```
proc sql;
  connect to sapiq as dbcon
    (host=iqsvr1 server=iqsvr1_users db=users user=iqusrl
     password=iqpwd1);
  select * from connection to dbcon
    (select * from customers where customer like '1%');
quit;
```

---

## Requirement for ORDER BY Variables

The SAP IQ database requires that a variable that is specified in an ORDER BY clause must also be included in a SELECT clause for a query. If you specify a variable in an ORDER BY clause that is not in the SELECT statement, you receive an error in the SAS log and the query fails.

---

## Special Catalog Queries

SAS/ACCESS Interface to SAP IQ supports the following special queries. You can use the queries to call the ODBC-style catalog function application programming interfaces (APIs). Here is the general format of the special queries:

`SIQ::SQLAPI "parameter 1","parameter n"`

`SIQ::`

is required to distinguish special queries from regular queries. `SIQ::` is not case sensitive.

`SQLAPI`

is the specific API that is being called. `SQLAPI` is not case sensitive.

`"parameter n"`

a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, this call to SQLTables usually matches table names such as myatest and my\_test:

```
select * from connection to sapiq (SIQ::SQLTables "test","","my_test");
```

Use the escape character to search only for the my\_test table:

```
select * from connection to sapiq (SIQ::SQLTables "test","","my\_test");
```

SAS/ACCESS Interface to SAP IQ supports these special queries.

`SIQ::SQLTables <"Catalog", "Schema", "Table-name", "Type">`

returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.

`SIQ::SQLColumns <"Catalog", "Schema", "Table-name", "Column-name">`

returns a list of all columns that match the specified arguments. If you do not specify any argument, all accessible column names and information are returned.

`SIQ::SQLPrimaryKeys <"Catalog", "Schema", "Table-name">`

returns a list of all columns that compose the primary key that matches the specified table. A primary key can be composed of one or more columns. If you do not specify any table name, this special query fails.

SIQ::SQLSpecialColumns <"*Identifier-type*", "*Catalog-name*", "*Schema-name*", "*Table-name*", "*Scope*", "*Nullable*">  
 returns a list of the optimal set of columns that uniquely identify a row in the specified table.

SIQ::SQLStatistics <"*Catalog*", "*Schema*", "*Table-name*">  
 returns a list of the statistics for the specified table name. You can set SQL\_INDEX\_ALL and SQL\_ENSURE options in the SQLStatistics API call. If you do not specify any table name argument, this special query fails.

SIQ::SQLGetTypeInfo  
 returns information about the data types that the SAP IQ database supports.

---

## Autopartitioning Scheme for SAP IQ

### Overview

Autopartitioning for SAS/ACCESS Interface to SAP IQ is a modulo (MOD) function method. For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page 76.

### Autopartitioning Restrictions

SAS/ACCESS Interface to SAP IQ places additional restrictions on the columns that you can use for the partitioning column during the autopartitioning phase. Here is how columns are partitioned.

- INTEGER, SMALLINT, and TINYINT columns are given preference.
- You can use DECIMAL, DOUBLE, FLOAT, NUMERIC, or NUMERIC columns for partitioning if the precision minus the scale of the column is greater than 0 but less than 10—namely,  $0 < (\text{precision-scale}) < 10$ .

### Nullable Columns

If you select a nullable column for autopartitioning, the OR<*column-name*>IS NULL SQL statement is appended at the end of the SQL code that is generated for the threaded Read. This ensures that any possible NULL values are returned in the result set. Also, if the column to be used for partitioning is specified as BIT, the number of threads are automatically changed to two, regardless how DBSLICEPARM= is set.

---

## Using WHERE Clauses

Autopartitioning does not select a column to be the partitioning column if it appears in a SAS WHERE clause. For example, this DATA step cannot use a threaded Read to retrieve the data because all numeric columns in the table are in the WHERE clause:

```
data work.locemp;
  set iqlib.MYEMPS;
  where EMPNUM<=30 and ISTENURE=0 and
        SALARY<=35000 and NUMCLASS>2;
run;
```

---

## Using DBSLICEPARM=

Although SAS/ACCESS Interface to SAP IQ defaults to three threads when you use autopartitioning, do not specify a maximum number of threads for the threaded Read in the [DBSLICEPARM= LIBNAME option on page 226](#).

---

## Using DBSLICE=

You might achieve the best possible performance when using threaded Reads by specifying the [DBSLICE= data set option](#) for SAP IQ in your SAS operation. This is especially true if you specified an index on one of the columns in the table. SAS/ACCESS Interface to SAP IQ selects only the first integer-type column in the table. This column might not be the same column where the index is specified. If so, you can specify the indexed column using DBSLICE=, as shown in this example.

```
proc print data=iqlib.MYEMPS(DBSLICE=( "EMPNUM BETWEEN 1 AND 33"
   "EMPNUM BETWEEN 34 AND 66" "EMPNUM BETWEEN 67 AND 100"));
run;
```

Using DBSLICE= also gives you flexibility in column selection. For example, if you know that the STATE column in your employee table contains only a few distinct values, you can customize your DBSLICE= clause accordingly.

```
datawork.locemp;
  set iqlib2.MYEMP(DBSLICE=( "STATE='FL'" "STATE='GA'"
                             "STATE='SC'" "STATE='VA'" "STATE='NC'" ));
  where EMPNUM<=30 and ISTENURE=0 and SALARY<=35000 and NUMCLASS>2;
run;
```

# Temporary Table Support for SAP IQ

SAS/ACCESS Interface to SAP IQ supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## Passing SAS Functions to SAP IQ

SAS/ACCESS Interface to SAP IQ passes the following SAS functions to SAP IQ for processing. Where the SAP IQ function name differs from the SAS function name, the SAP IQ name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                 |                    |
|-----------------|--------------------|
| ABS             | MIN                |
| ACOS (ACOS)     | MINUTE             |
| ARSIN (ASIN)    | MOD (see note)     |
| ATAN            | MONTH              |
| Avg             | QTR (QUARTER)      |
| BYTE (CHAR)     | REPEAT             |
| CEIL            | SECOND             |
| COALESCE        | SIGN               |
| COS             | SIN                |
| COUNT           | SQRT               |
| DAY             | STRIP (TRIM)       |
| EXP             | SUBSTR (SUBSTRING) |
| FLOOR           | SUM                |
| HOUR            | TAN                |
| INDEX (LOCATE)  | TRANWRD (REPLACE)  |
| LOG             | TRIMN (RTRIM)      |
| LOG10           | UPCASE (UPPER)     |
| LOWCASE (LOWER) | WEEKDAY (DOW)      |
| MAX             | YEAR               |

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also

pass these SAS SQL functions to SAP IQ. Due to incompatibility in date and time functions between SAP IQ and SAS, SAP IQ might not process them correctly. Check your results to determine whether these functions are working as expected.

|                              |                      |
|------------------------------|----------------------|
| COMPRESS (REPLACE)           | SOUNDEX              |
| DATE (CURRENT_DATE)          | TIME (CURRENT_TIME)  |
| DATEPART (DATE)              | TIMEPART (TIME)      |
| DATETIME (CURRENT_TIMESTAMP) | TODAY (CURRENT_DATE) |
| LENGTH (BYTE_LENGTH)         | TRIM                 |

---

## Passing Joins to SAP IQ

For a multiple libref join to pass to SAP IQ, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)
- host (HOST=)
- server (SERVER=)
- database (DATABASE=)
- port (PORT=)
- data source (DSN=, if specified)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Bulk Loading for SAP IQ

---

### Loading

Bulk loading is the fastest way to insert large numbers of rows into an SAP IQ table. You must specify **BULKLOAD=YES** to use the bulk-load facility. The bulk-load facility uses the SAP IQ LOAD TABLE command to move data from the client to the SAP IQ database.

## Data Set Options for Bulk Loading

Here are the SAP IQ bulk-load data set options.

- [BL\\_CLIENT\\_DATAFILE=](#)
- [BL\\_COLUMN\\_DELIMITER=](#) (alias [BL\\_DELIMITER=](#))
- [BL\\_DATAFILE=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_OPTIONS=](#)
- [BL\\_ROW\\_DELIMITER=](#)
- [BL\\_SERVER\\_DATAFILE=](#)
- [BL\\_USE\\_PIPE=](#)
- [BULKLOAD=](#)

## Bulk Loading Examples

In this example, the SASFLT.FLT98 SAS data set creates and loads FLIGHTS98, a large SAP IQ table. For SAP IQ 12.x, this works only when the SAP IQ server is on the same server as your SAS session.

```
libname sasflt 'SAS-library';
libname mydblib sapiq host=iqsrv1 server=iqsrv1_users
    db=users user=iqusrl password=iqpwd1;

proc sql;
create table mydblib.flights98
    (bulkload=YES)
        as select * from sasflt.flt98;
quit;
```

When the SAP IQ server and your SAS session are not on the same server, you need to include additional options, as shown in this example.

```
libname sasflt 'SAS-library';
libname mydblib sapiq host=iqsrv1 server=iqsrv1_users
    db=users user=iqusrl password=iqpwd1;
proc sql;
create table mydblib.flights98
( BULKLOAD=YES
    BL_USE_PIPE=NO
    BL_SERVER_DATAFILE='/tmp/fltdata.dat'
    BL_CLIENT_DATAFILE='/tmp/fltdata.dat' )
as select * from sasflt.flt98;
quit;
```

In this example, you can append the SASFLT.FLT98 SAS data set to the existing SAP IQ table, ALLFLIGHTS. The [BL\\_USE\\_PIPE=NO](#) option forces SAS/ACCESS

Interface to SAP IQ to write data to a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the load has completed.

```
proc append base=mydblib.allflights
  (BULKLOAD=YES
   BL_DATAFILE='/tmp/fltdata.dat'
   BL_USE_PIPE=NO
   BL_DELETE_DATAFILE=NO)
  data=sasflt.flt98;
  run;
```

## Locking in the SAP IQ Interface

These LIBNAME and data set options let you control how the SAP IQ interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134.

`READ_LOCK_TYPE= ROW | TABLE`

`UPDATE_LOCK_TYPE= ROW | TABLE`

`READ_ISOLATION_LEVEL= S | RR | RC | RU`

SAP IQ supports the S, RR, RC, and RU isolation levels that are defined in this table.

*Table 36.3 Isolation Levels for SAP IQ*

| Isolation Level       | Definition                                                                   |
|-----------------------|------------------------------------------------------------------------------|
| S (serializable)      | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.           |
| RR (repeatable read)  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads. |
| RC (read committed)   | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads. |
| RU (read uncommitted) | Allows dirty Reads, nonrepeatable Reads, and phantom Reads.                  |

Here are how the terms in the table are defined.

### *Dirty reads*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it can see changes that are made by those concurrent transactions even before they commit.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

*Nonrepeatable reads*

If a transaction exhibits this phenomenon, it might read a row once and later fail when it attempts to read that row again in the same transaction. The row might have been changed or even deleted by a concurrent transaction. Therefore, the read is not necessarily repeatable.

For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

*Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist, a phantom.

`UPDATE_ISOLATION_LEVEL= S | RR | RC`

SAP IQ supports the S, RR, and RC isolation levels defined in the preceding table.

## Naming Conventions for SAP IQ

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. SAS/ACCESS Interface to SAP IQ supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less because SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view. For more information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) SAP IQ is not case sensitive, so all names default to lowercase.

SAP IQ objects include tables, views, and columns. They follow these naming conventions.

- A name must be from 1 to 128 characters long.
- A name must begin with a letter (A through Z), underscore (\_), at sign (@), dollar sign (\$), or number sign (#).
- Names are not case sensitive. For example, CUSTOMER and Customer are the same, but object names are converted to lowercase when they are stored in the

SAP IQ database. However, if you enclose a name in quotation marks, it is case sensitive.

- A name cannot be an SAP IQ reserved word, such as WHERE or VIEW.
- A name cannot be the same as another SAP IQ object that has the same type.

# Data Types for SAP IQ

## Overview

Every column in a table has a name and a data type. The data type tells SAP IQ how much physical storage to set aside for the column and the form in which the data is stored. This information includes information about SAP IQ data types, null and default values, and data conversions.

For more information about SAP IQ data types and to determine which data types are available for your version of SAP IQ, see your SAP IQ documentation.

SAS/ACCESS Interface to SAP IQ does not directly support any data types that are not listed below. Any columns using these types are read into SAS as character strings.

## Supported SAP IQ Data Types

Here are the data types that SAS/ACCESS Interface to SAP IQ supports:

- Character data:

|                          |                     |
|--------------------------|---------------------|
| CHAR( <i>n</i> )         | VARCHAR( <i>n</i> ) |
| LONG VARCHAR( <i>n</i> ) |                     |

- Numeric data:

|                           |          |
|---------------------------|----------|
| BIGINT                    | INTEGER  |
| BIT                       | REAL     |
| DECIMAL   DEC   NUMERIC   | SMALLINT |
| DOUBLE   DOUBLE PRECISION | TINYINT  |
| FLOAT                     |          |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the

data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Date, Time, and Timestamp data:

**Note:** SQL date and time data types are collectively called datetime values. The SQL data types for dates, times, and timestamps are listed here. Be aware that columns of these data types can contain data values that are out of range for SAS.

DATE    TIMESTAMP  
TIME

## SAP IQ Null Values

SAP IQ has a special value called NULL. An SAP IQ NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads an SAP IQ NULL value, it interprets it as a SAS missing value.

To indicate that NULL values are not allowed, you can specify in SQL that a column is NOT NULL. The NOT NULL designation tells SQL to reject any rows that contains NULL data values for those columns. For example, if you designate the Score column as NOT NULL for a table, then any attempt to load a row without a value for Score results in an error. When you create a table with SAS/ACCESS, you can use the DBNULL= data set option to indicate whether NULL is a valid value for specified columns.

You can also define SAP IQ columns as NOT NULL DEFAULT. For more information about using the NOT NULL DEFAULT value, see your SAP IQ documentation.

For your data, determine whether an SAP IQ column allows NULL values or the host system supplies a default value for a column that is specified as NOT NULL WITH DEFAULT. Knowing this information, you can write selection criteria and enter values to update a table. Unless a column is specified as NOT NULL or NOT NULL DEFAULT, it allows NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how SAS missing character values are handled, use the **NULCHAR=** and **NULCHARVAL=** data set options.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to SAP IQ assigns to SAS variables when using the **LIBNAME statement** to read from an SAP IQ table. These default formats are based on SAP IQ column attributes.

**Table 36.4 LIBNAME Statement: Default SAS Formats for SAP IQ Data Types**

| SAP IQ Data Type                                                               | SAS Data Type | Default SAS Format      |
|--------------------------------------------------------------------------------|---------------|-------------------------|
| CHAR( <i>n</i> ) <sup>1</sup>                                                  | character     | \$ <i>w</i> .           |
| VARCHAR( <i>n</i> ) <sup>1</sup>                                               | character     | \$ <i>w</i> .           |
| LONG VARCHAR( <i>n</i> ) <sup>1</sup>                                          | character     | \$ <i>w</i> .           |
| BIGINT                                                                         | numeric       | 20.                     |
| SMALLINT                                                                       | numeric       | 6.                      |
| TINYINT                                                                        | numeric       | 4.                      |
| INTEGER                                                                        | numeric       | 11.                     |
| BIT                                                                            | numeric       | 1.                      |
| DOUBLE                                                                         | numeric       | none                    |
| REAL                                                                           | numeric       | none                    |
| FLOAT                                                                          | numeric       | none                    |
| DECIMAL( <i>p,s</i> ) <sup>2</sup>                                             | numeric       | <i>w.d</i>              |
| NUMERIC(10,4)                                                                  | numeric       | DOLLAR10.4 <sup>3</sup> |
| NUMERIC(19,4)                                                                  | numeric       | DOLLAR19.4 <sup>3</sup> |
| NUMERIC( <i>p,s</i> ) <sup>2</sup><br>(other than (10,4) or (19,4)<br>formats) | numeric       | <i>w.d</i>              |
| TIME                                                                           | numeric       | TIME8.                  |
| DATE                                                                           | numeric       | DATE9.                  |
| TIMESTAMP                                                                      | numeric       | DATETIME25.6            |

<sup>1</sup> The *n* in SAP IQ character data types is equivalent to *w* in SAS formats.<sup>2</sup> If a conflict might occur between the SAP IQ and SAS value for this data type, use **SASDATEFMT=** to specify the SAS format.<sup>3</sup> To change to a different format, use the **DBSASTYPE=** data set option to specify an alternative.

This table shows the default SAP IQ data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 36.5 LIBNAME Statement: Default SAP IQ Data Types for SAS Variable Formats**

| SAS Variable Format                                                                   | SAP IQ Data Type              |
|---------------------------------------------------------------------------------------|-------------------------------|
| w.d                                                                                   | DECIMAL( $p,s$ ) <sup>2</sup> |
| DOLLAR19.4<br>(14 digits or fewer for the dollar value and up to 4 fractional digits) | MONEY                         |
| DOLLAR10.4<br>(6 digits or fewer for the dollar value and up to 4 fractional digits)  | SMALLMONEY                    |
| other numerics                                                                        | DOUBLE                        |
| \$w                                                                                   | VARCHAR( $n$ ) <sup>1</sup>   |
| datetime formats                                                                      | TIMESTAMP                     |
| date formats                                                                          | DATE                          |
| time formats                                                                          | TIME                          |

**1**  $n$  in SAP IQ data types is equivalent to  $w$  in SAS formats.

**2**  $p$  and  $s$  in SAP IQ numeric data types are equivalent to  $w$  and  $d$  in SAS formats.

# SAS/ACCESS Interface to Snowflake

---

|                                                                        |      |
|------------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to Snowflake</i> ..... | 1272 |
| <i>Introduction to SAS/ACCESS Interface to Snowflake</i> .....         | 1272 |
| <i>LIBNAME Statement for the Snowflake Engine</i> .....                | 1272 |
| Overview .....                                                         | 1272 |
| Arguments .....                                                        | 1273 |
| Snowflake LIBNAME Statement Examples .....                             | 1277 |
| <i>Data Set Options for Snowflake</i> .....                            | 1278 |
| <i>SQL Pass-Through Facility Specifics for Snowflake</i> .....         | 1280 |
| Key Information .....                                                  | 1280 |
| CONNECT Statement Examples .....                                       | 1281 |
| <i>Understanding Snowflake Update and Delete Rules</i> .....           | 1282 |
| <i>Temporary Table Support for Snowflake</i> .....                     | 1283 |
| <i>Passing SAS Functions to Snowflake</i> .....                        | 1283 |
| <i>Passing Joins to Snowflake</i> .....                                | 1284 |
| <i>Bulk Loading and Unloading for Snowflake</i> .....                  | 1285 |
| Overview .....                                                         | 1285 |
| Data Set Options for Bulk Loading and Bulk Unloading .....             | 1286 |
| Examples .....                                                         | 1287 |
| <i>Naming Conventions for Snowflake</i> .....                          | 1288 |
| <i>Data Types for Snowflake</i> .....                                  | 1289 |
| Overview .....                                                         | 1289 |
| Supported Snowflake Data Types .....                                   | 1289 |
| LIBNAME Statement Data Conversions .....                               | 1290 |

---

# System Requirements for SAS/ACCESS Interface to Snowflake

You can find information about system requirements for SAS/ACCESS Interface to Snowflake in these locations.

- [System Requirements for SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements for Use with SAS 9.4](#)

---

# Introduction to SAS/ACCESS Interface to Snowflake

For available SAS/ACCESS features, see [Snowflake supported features](#). For more information about Snowflake, see your Snowflake documentation.

SAS/ACCESS Interface to Snowflake includes SAS Data Connector to Snowflake. The data connector enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections.

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- [“Snowflake Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)

---

# LIBNAME Statement for the Snowflake Engine

---

## Overview

This section describes the LIBNAME statement options that SAS/ACCESS Interface to Snowflake supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Snowflake.

**LIBNAME libref snow <connection-options> <LIBNAME-options>;**

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in *SAS Global Statements: Reference*.

## Arguments

### libref

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### snow

specifies the SAS/ACCESS engine name for the Snowflake interface.

### connection-options

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the Snowflake database in several ways. Specify only one of these required methods for each connection because they are mutually exclusive.

- SERVER=, PORT=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD=

You can also specify any of these optional connection options.

- DATABASE=
- ROLE=
- SCHEMA=
- WAREHOUSE=

The ODBC driver supports some additional connection options. To add these to the connection string for the SAS/ACCESS engine, specify them on the CONOPTS= LIBNAME option.

Here is how these options are defined.

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

### CONOPTS=<'>*Snowflake-additional-connection-options*<'>

specifies additional connection options for the Snowflake database. Separate multiple options with a semicolon. See your Snowflake ODBC driver documentation for a list of ODBC connection options that your ODBC driver supports.

Alias: CONNECT\_OPTIONS=

### DATABASE=<'>*database-name*<'>

specifies the default Snowflake database to use for sessions that the driver initiates and that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**Alias:** DB=

Requirement: If you specify this option, you must also set SCHEMA=.

**DRIVER=**

specifies the ODBC driver.

Default: SnowflakeDSIDriver

Requirement: On UNIX, this must be configured in odbcinst.ini.

**DSN=<'>Snowflake-data-source<'>**

specifies the configured Snowflake ODBC data source to which you want to connect. Use this option if you have existing Snowflake ODBC data sources that are configured on your client. This method requires additional setup through the odbc.ini file on UNIX platforms. So it is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

**Alias:** DATASRC=, DS=

Requirement: If you use this method, you just specify the user ID and password in the LIBNAME statement, even if these are already specified in the ODBC data source.

**PASSWORD=<'>Snowflake-password<'>**

**Alias:** PASS=, PW=, PWD=, USING=

specifies the password that is associated with your Snowflake user ID. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**PORt=port**

specifies the port number that is used to connect to the specified Snowflake database or server.

**Alias:** SERVICE=, SERVICE\_NAME=

Default: 443

**ROLE=<'>Snowflake-role<'>**

specifies the default Snowflake role to use for sessions that the driver initiated. The specified role should be one that has been assigned to the specified user. If the specified role does not match any roles that are assigned to the user, sessions that the driver initiated have no role initially. However, a role can always be specified from within the session.

**SCHEMA=<'>Snowflake-schema<'>**

specifies the default Snowflake schema to use for sessions that the driver initiated.

**Alias:** OWNER=

Default: PUBLIC

Requirement: The SCHEMA= option is required in LIBNAME connections to Snowflake.

**SERVer=<'>server-name<'>**

specifies the host name or IP address where the Snowflake database is running. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**Alias:** HOST=, HOSTNAME=

**USER=<'>Snowflake-user-name<'>**

specifies the Snowflake user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: **UID=**

**WAREHOUSE=<'>Snowflake-warehouse<'>**

specifies the default Snowflake warehouse to use for sessions that the driver initiated.

#### *LIBNAME options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Snowflake with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 37.1** SAS/ACCESS LIBNAME Options for Snowflake

| Option                       | Default Value | Valid in CONNECT |
|------------------------------|---------------|------------------|
| <b>ACCESS=</b>               | none          |                  |
| <b>AUTHDOMAIN=</b>           | none          |                  |
| <b>AUTOCOMMIT=</b>           | NO            | •                |
| <b>BULKLOAD=</b>             | none          |                  |
| <b>BULKUNLOAD=</b>           | NO            |                  |
| <b>CONNECTION=</b>           | SHAREDREAD    | •                |
| <b>CONNECTION_GROUP =</b>    | none          | •                |
| <b>CONOPTS=</b>              | none          | •                |
| <b>DBCLIENT_MAX_BYTES =</b>  | 0             | •                |
| <b>DBCOMMIT=</b>             | none          |                  |
| <b>DBCONINIT=</b>            | none          | •                |
| <b>DBCONTERM=</b>            | none          | •                |
| <b>DBCREATE_TABLE_OP_TS=</b> | none          |                  |
| <b>DBGEN_NAME=</b>           | DBMS          | •                |

| Option                  | Default Value                                      | Valid in CONNECT |
|-------------------------|----------------------------------------------------|------------------|
| DBLIBINIT=              | none                                               |                  |
| DBLIBTERM=              | none                                               |                  |
| DBMAX_TEXT=             | 1024                                               | •                |
| DBMSTEMP=               | NO                                                 |                  |
| DBNULLKEYS=             | YES                                                |                  |
| DBPROMPT=               | NO                                                 | •                |
| DBSASLABEL=             | COMPAT                                             |                  |
| DBSERVER_MAX_BYTE<br>S= | 1                                                  |                  |
| DEFER=                  | NO                                                 | •                |
| DELETE_MULT_ROWS=       | NO                                                 |                  |
| DIRECT_SQL=             | YES                                                |                  |
| INSERTBUFF=             | automatically calculated<br>based on row length    |                  |
| LOGIN_TIMEOUT=          | 0                                                  | •                |
| MULTI_DATASRC_OPT=      | NONE                                               |                  |
| POST_STMT_OPTS=         | none                                               |                  |
| PRESERVE_COL_NAME<br>S= | YES (see “Naming<br>Conventions for<br>Snowflake”) |                  |
| PRESERVE_TAB_NAME<br>S= | YES (see “Naming<br>Conventions for<br>Snowflake”) |                  |
| QUERY_TIMEOUT=          | 0                                                  | •                |
| READBUFF=               | automatically calculated<br>based on row length    | •                |
| REREAD_EXPOSURE=        | NO                                                 | •                |
| SCHEMA=                 | PUBLIC                                             |                  |

| Option                  | Default Value | Valid in CONNECT |
|-------------------------|---------------|------------------|
| SPOOL=                  | YES           |                  |
| SQL_FUNCTIONS=          | none          |                  |
| SQL_FUNCTIONS_COP<br>Y= | none          |                  |
| SQLGENERATION=          | none          |                  |
| STRINGDATES=            | NO            |                  |
| TRACE=                  | NO            |                  |
| TRACEFILE=              | none          | •                |
| UPDATE_MULT_ROWS=       | NO            |                  |
| UTILCONN_TRANSIENT<br>= | NO            |                  |

## Snowflake LIBNAME Statement Examples

In the example below, SERVER=, DATABASE=, USER=, and PASSWORD= are the connection options.

```
LIBNAME mydblib snow server=mysrv1 database=test
      user=myusr1 password=mypwd1;
proc print data=mydblib.customers;
      where state='CA';
run;
```

In this next example, DSN=, USER=, and PASSWORD= are the connection options. The Snowflake database is configured in the odbc.ini file or a similarly named configuration file on UNIX platforms.

```
LIBNAME mydblib snow dsn=snow user=myusr1 password=mypwd1
      schema=sasuser;

proc print data=mydblib.customers;
      where state='CA';
run;
```

In the example below, SERVER=, USER=, PASSWORD=, and SCHEMA= are the connection options.

```
libname a snow server="saspartner.snowflakecomputing.com"
      user=myusr1 password=mypwd1 schema=sasuser;
```

# Data Set Options for Snowflake

All SAS/ACCESS data set options in this table are supported for Snowflake. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 37.2** SAS/ACCESS Data Set Options for Snowflake

| Option                   | Default Value                                                           |
|--------------------------|-------------------------------------------------------------------------|
| BL_AWS_CONFIG_FILE=      | ~/.aws/config or \\aws\\config                                          |
| BL_AWS_PROFILE_NAME=     | none                                                                    |
| BL_AWS_CREDENTIALS_FILE= | ~/.aws/credentials                                                      |
| BL_BUCKET=               | none                                                                    |
| BL_COMPRESS=             | NO                                                                      |
| BL_CONFIG=               | none                                                                    |
| BL_DEFAULT_DIR=          | temporary file directory that the UTILLOC= system option specifies      |
| BL_DELETE_DATAFILE=      | YES                                                                     |
| BL_DELIMITER=            | 0x07 (ASCII character)                                                  |
| BL_ENCKEY=               | none                                                                    |
| BL_FORMAT_OPT=           | NONE                                                                    |
| BL_INTERNAL_STAGE=       | none (see “ <a href="#">Bulk Loading and Unloading for Snowflake</a> ”) |
| BL_KEY=                  | none                                                                    |
| BL_NUM_DATAFILES=        | 2                                                                       |
| BL_NUM_READ_THREADS=     | none                                                                    |
| BL_OPTIONS=              | none                                                                    |
| BL_REGION=               | none                                                                    |

| Option               | Default Value                  |
|----------------------|--------------------------------|
| BL_SECRET=           | none                           |
| BL_TOKEN=            | none                           |
| BL_USE_ESCAPE=       | NO                             |
| BL_USE_SSL=          | none                           |
| BULKLOAD=            | none                           |
| BULKUNLOAD=          | NO                             |
| DBCOMMIT=            | LIBNAME option value           |
| DBCONDITION=         | none                           |
| DBCREATE_TABLE_OPTS= | LIBNAME option value           |
| DBFORCE=             | NO                             |
| DBGEN_NAME=          | DBMS                           |
| DBKEY=               | none                           |
| DBLABEL=             | NO                             |
| DBLARGETABLE=        | none                           |
| DBMAX_TEXT=          | 1024                           |
| DBNULL=              | YES                            |
| DBNULLKEYS=          | LIBNAME option value           |
| DBPROMPT=            | LIBNAME option value           |
| DBSASLABEL=          | COMPAT                         |
| DBSASTYPE=           | see “Data Types for Snowflake” |
| DBTYPE=              | see “Data Types for Snowflake” |
| ERRLIMIT=            | 1                              |
| INSERTBUFF=          | LIBNAME option value           |
| NULCHAR=             | SAS                            |

| Option              | Default Value        |
|---------------------|----------------------|
| NULLCHARVAL=        | a blank character    |
| POST_STMT_OPTS=     | none                 |
| POST_TABLE_OPTS=    | none                 |
| PRE_STMT_OPTS=      | none                 |
| PRE_TABLE_OPTS=     | none                 |
| PRESERVE_COL_NAMES= | LIBNAME option value |
| QUERY_TIMEOUT=      | LIBNAME option value |
| READBUFF=           | LIBNAME option value |
| SASDATEFMT=         | none                 |
| SCHEMA=             | LIBNAME option value |

---

## SQL Pass-Through Facility Specifics for Snowflake

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Snowflake interface.

- The *dbms-name* is `snow`.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Snowflake. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default `snow` alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).
- Snowflake does not support indexes. Statements that pertain to indexes, such as the CREATE INDEX statement, are not supported.

**IMPORTANT** Issuing multiple INSERT commands within PROC SQL or PROC FEDSQL might result in slow performance. To insert multiple records, it is therefore recommended that you use a method that inserts them all at once, such as one these methods.

- a single INSERT command to insert multiple records
- a DATA step to copy multiple records from another source or from cards input
- bulk loading, when available

**Note:** A FedSQL INSERT statement can insert values for only one row at a time. To insert multiple rows, you need multiple INSERT statements.

## CONNECT Statement Examples

This example connects to Snowflake and then disconnects from it.

```
proc sql noerrorstop;
  connect to snow as x1(server=mysrv1 port=2406
    user=mysurl password='mypwd1' database=mydb1);
  disconnect from x1;
  quit;
```

This next example connects to Snowflake, executes some SQL statements, and then disconnects from Snowflake.

```
proc sql noerrorstop;
  connect to snow as x1(server=mysrv1 port=2406
    user=mysurl password='mypwd1' database=mydb1);

  execute ( CREATE TABLE t1 ( no int primary key, state varchar(10) ) )
  by x1;
  execute ( INSERT INTO t1 values (1, 'USA') ) by x1;
  execute ( INSERT INTO t1 values (2, 'CHN') ) by x1;
  select * from connection to x1 (SELECT * FROM t1 ORDER BY no);

  disconnect from x1;
  quit;
```

The Snowflake engine can process CREATE TABLE *table-name* AS SELECT in a single step. Using the DBIDIRECTEXEC system option, the SQL procedure can pass CREATE TABLE AS SELECT statements to Snowflake. By default, the system option is specified for the Snowflake engine. In this example, the schema is specified in the DSN.

```
options sastrace=',,,d' sastraceloc=saslog nostsuffix msglevel=i;
libname x snow dsn='snowflake' user=sasuser pw=XXXXXXXXX
schema=myschema;

proc delete data=x.class1; run;
proc delete data=x.class2; run;
```

```

data x.class1;
  set sashelp.class;
run;

proc sql;
  create table x.class2 as select * from x.class1;
quit;

```

Here is how the above code is passed.

```

CREATE TABLE "SCHEMA1"."class2" AS
  SELECT TXT_1."Name", TXT_1."Sex", TXT_1."Age", TXT_1."Height",
  TXT_1."Weight"
    FROM "SCHEMA1"."class1" TXT_1

```

If the NODBIDIRECTEXEC option is specified, the statement is executed in multiple steps, transferring the data to and from SAS.

```

CREATE TABLE "SCHEMA1"."class2"
  ("Name" VARCHAR(8), "Sex" VARCHAR(1), "Age" DOUBLE, "Height"
  DOUBLE, "Weight" DOUBLE)

INSERT INTO "SCHEMA1"."class2"
  ("Name", "Sex", "Age", "Height", "Weight")
VALUES ( ?, ?, ?, ?, ?, ? )

```

## Understanding Snowflake Update and Delete Rules

To avoid data integrity problems when updating or deleting data, you must specify a primary key on your table. This example uses DBTYPE= to create the primary key.

```

libname invty snow user=myusr1 server=mysrv1 database=test reread_exposure=no;

proc sql;
drop table invty STOCK23;
quit;

/* Create DBMS table with primary key */
data invty.STOCK23(drop=PARTNO DBTYPE=(RECDATE="date not null,
  primary key(RECDATE")));
  input PARTNO $ DESCX $ INSTOCK @20
    RECDATE date7. @29 PRICE;
  format RECDATE date7.;
  datalines;
K89R seal      34 27jul95 245.00
M447 sander    98 20jun95 45.88
LK43 filter    121 19may96 10.99
MN21 brace     43 10aug96 27.87
BC85 clamp     80 16aug96  9.55
KJ66 cutter     6 20mar96 24.50

```

```

UYN7    rod      211  18jun96   19.77
JD03    switch   383  09jan97   13.99
BV1I    timer     26   03jan97   34.50
;

```

This next example shows how you can update the table now that STOCK23 has a primary key.

```

proc sql;
update invty STOCK23 set price=price*1.1 where INSTOCK > 50;
quit;

```

## Temporary Table Support for Snowflake

SAS/ACCESS Interface to Snowflake supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

## Passing SAS Functions to Snowflake

SAS/ACCESS Interface to Snowflake passes the following SAS functions to Snowflake for processing. When the Snowflake function name differs from the SAS function name, the Snowflake name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                           |                            |
|---------------------------|----------------------------|
| ABS                       | LOG10 (LOG(10, <i>n</i> )) |
| ARCOS (ACOS)              | LOG2 (LOG(2, <i>n</i> ))   |
| ARSIN (ASIN)              | LOWCASE (LOWER)            |
| ATAN                      | MINUTE                     |
| ATAN2                     | MOD                        |
| CAT (CONCAT)              | MONTH                      |
| CEIL                      | QTR (QUARTER)              |
| COALESCE                  | REPEAT                     |
| COS                       | SECOND                     |
| COSH                      | SIGN                       |
| COT                       | SIN                        |
| DAY (DAYOFMONTH)          | SINH                       |
| DTEXTDAY (DAYOFMONTH)     | SQRT                       |
| DTEXTMONTH (MONTH)        | STD (STDDEV)               |
| DTEXTYEAR (YEAR)          | STRIP (TRIM)               |
| DTEXTWEEKDAY (DAYOFWEEK)* | SUBSTR                     |
| EXP                       | TAN                        |
| FLOOR                     | TANH                       |

|                                         |                             |
|-----------------------------------------|-----------------------------|
| HOUR                                    | TRANWRD<br>(REGEXP_REPLACE) |
| INDEX (CHARINDEX)                       | TRIMN (RTRIM)               |
| LEFT (LTRIM)                            | UPCASE (UPPER)              |
| LENGTH (OCTET_LENGTH(RTRIM()))**<br>*** | VAR (VARIANCE)              |
| LENGTHC (LENGTH)                        | WEEKDAY (DAYOFWEEK)*        |
| LOG (LN)                                | YEAR                        |

\*At the start of a session, WEEK\_START is set to 7.

\*\*Returns the length of a string or binary value in bytes. This is the same as LENGTH for ASCII strings and greater than LENGTH for strings that use Unicode code points. For binary, this is always the same as LENGTH.

\*\*\*Snowflake does not support passing binary columns to length functions.

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when SQL\_FUNCTIONS=ALL can the SAS/ACCESS engine also pass these SAS SQL functions to Snowflake. Because of the incompatibility in date and time functions between Snowflake and SAS, Snowflake might not process them correctly. Check your results to determine whether these functions are working as expected.

|                                |                        |
|--------------------------------|------------------------|
| COMPRESS (TRANSLATE)*          | TIME (CURRENT_TIME)**  |
| DATE (CURRENT_DATE)**          | TIMEPART (TO_TIME)     |
| DATEPART (TO_DATE)             | TODAY (CURRENT_DATE)** |
| DATETIME (CURRENT_TIMESTAMP)** |                        |

\*Only COMPRESS with one or two arguments is passed to Snowflake. Because the third argument is not supported for pass-down, SAS always handles it.

\*\*Set TIMEZONE= in the DSN= connection option (for example, TIMEZONE=America/New\_York).

---

## Passing Joins to Snowflake

In order for a multiple libref join to pass to Snowflake, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- database (DATABASE=)

- port (PORT=)
- SQL functions (SQL\_FUNCTIONS=)
- initialization commands (DBCONINIT=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

# Bulk Loading and Unloading for Snowflake

---

## Overview

Bulk loading is the fastest way to insert large numbers of rows into a Snowflake table. To use the bulk-load facility, set the **BULKLOAD=** data set option to YES. You can also perform bulk unloading (data retrieval) from Snowflake. To enable bulk unloading of data, set the **BULKUNLOAD= LIBNAME** option to YES.

The SAS/ACCESS engine handles bulk loading and bulk unloading through one of these methods.

- Snowflake internal stage: The engine uses PUT and GET commands to upload files to the internal stage. Bulk loading and unloading use the Snowflake COPY command to load staged files into or from a table.
- Amazon S3 bucket: The bulk-load facility uses the Amazon Simple Storage Service (S3) tool to move data to and from the client to the Snowflake database. Files are copied using S3 utilities. You can find more information about managing data on Snowflake on the [Amazon Web Services documentation site](#).

Snowflake supports different types of stages.

- user stages: @~ is the name of the stage of the current user.
- table stages: @[<namespace>.]%<tablename> is the name of the stage for a table.
- internally named stages: @[<namespace>.]<internal\_stage\_names> is the name of the internal stage.
- external stages: These are supported only through the Amazon S3 bucket. To specify the Amazon S3 location to use for bulk loading or bulk unloading, use these LIBNAME options or data set options.
  - BL\_AWS\_CONFIG\_FILE=
  - BL\_AWS\_PROFILE\_NAME=
  - BL\_BUCKET=
  - BL\_CONFIG\_FILE=
  - BL\_ENCKEY=
  - BL\_KEY=

- BL\_REGION=
- BL\_SECRET=
- BL\_TOKEN=
- BL\_USE\_SSL=

**Note:**

Bulk loading and bulk unloading use CSV files, which can be compressed.

User and table stages are created automatically. You can create internally named stages with the Snowflake CREATE STAGE command. For details, see your Snowflake documentation.

Use the BL\_INTERNAL\_STAGE= LIBNAME option or data set option to specify the stage to use for bulk loading or bulk unloading. USER and TABLE are used as placeholders for the respective stages. You can add a path to the location. The internal stage is used by the SAS/ACCESS engine. The engine does not create the stage.

---

## Data Set Options for Bulk Loading and Bulk Unloading

Here are the Snowflake bulk-load data set options. For detailed information about these options, see “[Overview](#)” on page 370. For bulk loading, you must use these data set options.

Except for BULKLOAD= and BULKUNLOAD=, all of these options are supported for bulk loading and bulk unloading.

**Table 37.3** Snowflake Data Set Options for Bulk Loading and Bulk Unloading

| Data Set Option      | Snowflake Internal Staging | Amazon External Location |
|----------------------|----------------------------|--------------------------|
| BL_AWS_CONFIG_FILE=  |                            | •                        |
| BL_AWS_PROFILE_NAME= |                            | •                        |
| BL_BUCKET=           |                            | •                        |
| BL_COMPRESS=         | •                          |                          |
| BL_CONFIG=           |                            | •                        |
| BL_DEFAULT_DIR=      | •                          |                          |
| BL_DELETE_DATAFILE=  | •                          |                          |

| Data Set Option      | Snowflake Internal Staging | Amazon External Location                  |
|----------------------|----------------------------|-------------------------------------------|
| BL_DELIMITER=        | •                          |                                           |
| BL_ENCKEY=           |                            | • (see “Encryption with Amazon Redshift”) |
| BL_INTERNAL_STAGE=   | •                          | •                                         |
| BL_KEY=              |                            | •                                         |
| BL_NUM_DATAFILES=    | •                          |                                           |
| BL_NUM_READ_THREADS= | •                          |                                           |
| BL_OPTIONS=          | •                          |                                           |
| BL_REGION=           |                            | •                                         |
| BL_SECRET=           |                            | •                                         |
| BL_TOKEN=            |                            | •                                         |
| BL_USE_ESCAPE=       | •                          |                                           |
| BL_USE_SSL=          |                            | •                                         |
| BULKLOAD=            | •                          | •                                         |
| BULKUNLOAD=          | •                          | •                                         |

## Examples

Here are some examples for various types of stages using BULKLOAD= and BULKUNLOAD=.

```
/* Uses the current user stage, @~/test1/ */
libname a snow <connection-options>
      bulkload=yes bulkunload=yes bl_internal_stage="user/test1";

/* Uses the table stage, @%"TABLE_NAME"/test1/, */
/* where TABLE_NAME is replaced with the table to load/unload */
libname a snow <connection-options>
      bulkload=yes bulkunload=yes bl_internal_stage="table/test1";

/* Uses the internal named stage, @MY_INTERNAL_STAGE/test1/ */
```

```
libname a snow <connection-options>
  bulkload=yes bulkunload=yes bl_internal_stage="MY_INTERNAL_STAGE/
  test1";
```

**Note:** If BL\_INTERNAL\_STAGE= is specified, options that support Amazon S3 buckets are ignored.

## Naming Conventions for Snowflake

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Snowflake interface supports table names and column names that contain up to 32 characters. If column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical column names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or fewer. SAS does not truncate a name longer than 32 characters. If you have a table name that is longer than 32 characters, it is recommended that you create a table view.

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Snowflake is not case sensitive, so all names default to lowercase.

Snowflake objects include tables, views, and columns. They follow these conventions.

- A name can contain up to 128 characters.
- The first character in a name can be a letter, @, \_, or #.
- A name must begin with a letter (A through Z), diacritic marks, non-Latin characters (200–377 octal) or an underscore (\_).
- To enable case sensitivity, enclose names in quotation marks. All references to quoted names must always be enclosed in quotation marks.
- A name cannot be a reserved word in Snowflake such as WHERE or VIEW.
- A name cannot be the same as another Snowflake object that has the same type.

# Data Types for Snowflake

## Overview

Every column in a table has a name and a data type. The data type tells Snowflake how much physical storage to set aside for the column and the form in which the data is stored. For more information about Snowflake data types and to determine which data types are available for your version of Snowflake, see your Snowflake documentation.

**IMPORTANT** The Snowflake ODBC driver limits the number of columns to 16,384 because of limitations for the length of SQL commands. The driver also limits the number of columns by the data length that is required to read or write one row. When all data is only numeric, this results in a limit of about 7,480 columns. The actual limit depends on the data types that are used and therefore might be lower.

For information about Snowflake data types and to determine which data types are available for your version of Snowflake, see your Snowflake documentation.

## Supported Snowflake Data Types

Here are the data types that the Snowflake engine supports.

■ Character data

|                 |            |
|-----------------|------------|
| ARRAY           | TEXT       |
| CHAR, CHARACTER | VARCHAR(n) |
| OBJECT          | VARIANT    |
| STRING          |            |

■ Numeric data

|                  |             |
|------------------|-------------|
| BIGINT           | FLOAT8      |
| BOOLEAN          | INT         |
| DECIMAL          | INTEGER     |
| DOUBLE           | NUMBER(p,s) |
| DOUBLE PRECISION | NUMERIC     |
| FLOAT            | REAL        |
| FLOAT4           | SMALLINT    |

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

- Date, time, and timestamp data

|           |               |
|-----------|---------------|
| DATE      | TIMESTAMP_LTZ |
| DATETIME  | TIMESTAMP_NTZ |
| TIME      | TIMESTAMP_TZ  |
| TIMESTAMP |               |

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Snowflake assigns to SAS variables when using the [LIBNAME statement](#) to read from a Snowflake table. These default formats are based on Snowflake column attributes. SAS/ACCESS does not support Snowflake data types that do not appear in this table.

*Table 37.4 LIBNAME Statement: Default SAS Formats for Snowflake Data Types*

| Snowflake Data Type | SAS Data Type | Default SAS Format                                                              |
|---------------------|---------------|---------------------------------------------------------------------------------|
| ARRAY               | character     | \$w., where w is the minimum of 32767 and the value of the DBMAX_TEXT= option   |
| BINARY(n)           | character     | \$HEXw.                                                                         |
| CHAR, CHARACTER     | character     | synonymous with VARCHAR except that the default length is VARCHAR(1)            |
| OBJECT              | character     | \$w., where w is the minimum of 32767 and the value of the DBMAX_TEXT= option** |
| STRING              | character     | synonymous with VARCHAR                                                         |

| Snowflake Data Type  | SAS Data Type | Default SAS Format                                                                                                                                |
|----------------------|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| TEXT                 | character     | synonymous with VARCHAR                                                                                                                           |
| VARCHAR( <i>n</i> )  | character     | \$ <i>w</i> .                                                                                                                                     |
| VARIANT              | character     | \$ <i>w</i> ., where <i>w</i> is the minimum of 32767 and the value of the DBMAX_TEXT= option**                                                   |
| BIGINT               | numeric       | synonymous with NUMBER                                                                                                                            |
| BOOLEAN              | numeric       | 1.                                                                                                                                                |
| DECIMAL              | numeric       | synonymous with NUMBER                                                                                                                            |
| DOUBLE               | numeric       | synonymous with FLOAT                                                                                                                             |
| DOUBLE PRECISION     | numeric       | synonymous with FLOAT                                                                                                                             |
| FLOAT                | numeric       | none<br>Snowflake uses double-precision (64-bit) IEEE 754 floating point numbers.                                                                 |
| FLOAT4               | numeric       | none<br>Snowflake uses double-precision (64-bit) IEEE 754 floating point numbers.                                                                 |
| FLOAT8               | numeric       | none<br>Snowflake uses double-precision (64-bit) IEEE 754 floating point numbers.                                                                 |
| INT                  | numeric       | synonymous with NUMBER                                                                                                                            |
| INTEGER              | numeric       | synonymous with NUMBER                                                                                                                            |
| NUMBER( <i>p,s</i> ) | numeric       | <i>w.d</i><br>If the precision or scale does not fit into SAS <i>w.d</i> format, use the default format, which means that no format is specified. |
| NUMERIC              | numeric       | synonymous with NUMBER                                                                                                                            |

| Snowflake Data Type | SAS Data Type | Default SAS Format                                                                                |
|---------------------|---------------|---------------------------------------------------------------------------------------------------|
| REAL                | numeric       | synonymous with FLOAT                                                                             |
| SMALLINT            | numeric       | synonymous with NUMBER                                                                            |
| DATE                | numeric       | DATE9.                                                                                            |
| DATETIME            | numeric       | DATETIME <i>w.d</i> , where <i>w</i> and <i>w</i> depend on precision<br>Alias for TIMESTAMP_NTZ. |
| TIME                | numeric       | TIME8.                                                                                            |
| TIMESTAMP           | numeric       | DATETIME <i>w.d</i> , where <i>w</i> and <i>w</i> depend on precision*                            |
| TIMESTAMP_LTZ       | numeric       | DATETIME <i>w.d</i> , where <i>w</i> and <i>w</i> depend on precision*                            |
| TIMESTAMP_NTZ       | numeric       | DATETIME <i>w.d</i> , where <i>w</i> and <i>w</i> depend on precision*                            |
| TIMESTAMP_TZ        | numeric       | DATETIME <i>w.d</i> , where <i>w</i> and <i>w</i> depend on precision*                            |
| VARBINARY           |               | synonymous with BINARY**                                                                          |

\*TIMESTAMP data types: SAS/ACCESS Interface for Snowflake sets the TIMESTAMP data type alias to map to TIMESTAMP without time zone. When the connection to Snowflake is established, these commands are executed.

```
ALTER SESSION SET TIMESTAMP_TYPE_MAPPING = TIMESTAMP_NTZ
ALTER SESSION SET CLIENT_TIMESTAMP_TYPE_MAPPING = TIMESTAMP_NTZ
ALTER SESSION SET WEEK_START=7
```

\*\*Semi-structured data types: The ODBC driver describes these as SQL\_VARCHAR, and they are read into SAS as a string. To read semi-structured data elements, use explicit SQL pass-through to submit SQL queries using special operators and functions.

Here is an example of semi-structured data.

```
libname snow dsn='snowflake' user=myusr1 pw=mypwd1
      schema=myschema;
proc sql noerrorstop;
connect using snow;
execute (
CREATE OR REPLACE TABLE "myClass" ( "src" variant )
AS
SELECT PARSE_JSON(column1) AS "src"
FROM VALUES
('{'
```

```

"Name" : "Alfred",
"Sex" : "M",
"Age" : 14,
"Measurements" : {
    "Height": 69.0,
    "Weight" : 112.5
}
}),
('{
    "Name" : "Alice",
    "Sex" : "F",
    "Age" : 13,
    "Measurements" : {
        "Height": 56.5,
        "Weight" : 84.0
    }
}) ) by snow;
disconnect from snow;
quit;

proc print data=snow.myClass;
title 'proc print myClass';
run;

title 'Traversing semi-structured data';
proc sql;
connect using snow;
select name, sex from connection to snow
(select "src":Name as name, "src":Sex as sex from
SASUSER."myClass");
disconnect from snow;
quit;

title 'Traversing semi-structured data - type specified';
proc sql;
connect using snow;
select name, sex from connection to snow
(select "src":Name::char(10) as name,
"src":Sex::char(1) as sex from SASUSER."myClass");
disconnect from snow;
quit;

```

The first PROC PRINT statement prints the semi-structured data in a string. The first SQL step uses the colon (:) notation to read a first-level element of the semi-structured data. In the second SQL step, a type specification is added.

```

proc print myClass          09:06 Thursday, November 22, 2018  2
  Obs src

    1 {
      "Age": 14,
      "Measurements": {
        "Height": 69,
        "Weight": 112.5
      },
      "Name": "Alfred",
      "Sex": "M"
    }
    2 {
      "Age": 13,
      "Measurements": {
        "Height": 56.5,
        "Weight": 84
      },
      "Name": "Alice",
      "Sex": "F"
    }
Traversing semi-structured data      09:06 Thursday, November 22, 2018  4

NAME           SEX
-----
"Alfred"       "M"
"Alice"        "F"

Traversing semi-structured data, type specified 09:06 Thursday, November 22, 2018  5

NAME           SEX
-----
Alfred         M
Alice          F

```

The next table shows the default Snowflake data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 37.5 LIBNAME Statement: Default Snowflake Data Types for SAS Variable Formats**

| SAS Data Type | SAS Variable Format | Snowflake Data Type  |
|---------------|---------------------|----------------------|
| character     | \$HEXw.             | BINARY( <i>n</i> )   |
| character     | \$w.                | VARCHAR( <i>n</i> )  |
| numeric       | datetime formats    | TIMESTAMP            |
| numeric       | date formats        | DATE                 |
| numeric       | time formats        | TIME                 |
| numeric       | w.d.                | NUMBER( <i>p,s</i> ) |

**SAS Data Type   SAS Variable Format   Snowflake Data Type**

numeric

other numerics

DOUBLE

- 
- 1 In a Snowflake data type,  $n$  is equivalent to  $w$  in SAS formats.



# SAS/ACCESS Interface to Spark

---

|                                                              |      |
|--------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to Spark</i> | 1298 |
| <i>Introduction to SAS/ACCESS Interface to Spark</i>         | 1298 |
| Overview                                                     | 1298 |
| Spark Concepts                                               | 1298 |
| <i>LIBNAME Statement for the Spark Engine</i>                | 1299 |
| Overview                                                     | 1299 |
| Arguments                                                    | 1299 |
| Connection Options for the Apache Hive JDBC Driver           | 1303 |
| Spark LIBNAME Statement Examples                             | 1304 |
| <i>Data Set Options for Spark</i>                            | 1305 |
| <i>SQL Pass-Through Facility Specifics for Spark</i>         | 1306 |
| Key Information                                              | 1306 |
| CONNECT Statement Examples                                   | 1307 |
| <i>Passing SAS Functions to Spark</i>                        | 1308 |
| <i>Passing Joins to Spark</i>                                | 1309 |
| <i>Bulk Loading for Spark</i>                                | 1309 |
| Overview                                                     | 1309 |
| Loading                                                      | 1309 |
| Examples                                                     | 1310 |
| <i>Naming Conventions for SAS and Spark</i>                  | 1311 |
| <i>Data Types for Spark</i>                                  | 1312 |
| Supported Spark Data Types                                   | 1312 |
| LIBNAME Statement Data Conversions                           | 1312 |
| Spark Null Values                                            | 1314 |
| SAS Table Properties for Hive and Spark                      | 1314 |
| <i>Sample Programs for Spark</i>                             | 1315 |
| Code Snippets                                                | 1315 |
| Use DBSASTYPE= to Load Spark Data into SAS                   | 1316 |
| Create a Partitioned Table with a File Type of SEQUENCEFILE  | 1317 |

---

# System Requirements for SAS/ACCESS Interface to Spark

You can find information about system requirements for SAS/ACCESS Interface to Spark in these locations:

- [System Requirements for SAS 9.4 Foundation for Linux for x64](#)
- [System Requirements for SAS/ACCESS Interface to Hadoop with SAS 9.4](#)
- [Third-Party Software Requirements](#)

See also: "Configuring SAS/ACCESS Interface to Spark" in [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#)

---

# Introduction to SAS/ACCESS Interface to Spark

---

## Overview

SAS/ACCESS Interface to Spark enables you to issue SparkSQL queries on data in Spark data sources, including Databricks.

For available SAS/ACCESS features, see [Spark supported features](#).

SAS/ACCESS Interface to Spark was implemented in [SAS 9.4M7](#). Databricks support starts in [SAS 9.4M8](#).

---

## Spark Concepts

Spark is a data processing environment suited to processing large-scale data.

Spark SQL is a Spark module that can act as a distributed SQL query engine. SAS/ACCESS to Spark generates Spark SQL queries to interact with Spark.

Databricks is a Spark data platform that runs in a hosted cloud environment, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure.

For more information about Spark, see the [Spark documentation](#). For more information about Databricks, see the [Databricks documentation](#).

---

# LIBNAME Statement for the Spark Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Spark supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

---

**Note:**

SAS 9.4M8 expands support for JDBC drivers other than the Apache Hive open source driver. SAS supports the Databricks JDBC Driver for access to Spark data in Databricks data sources. For more information, see ["Configuring SAS/ACCESS Interface to Spark" in SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#). Connections using third-party drivers are supported through the URI= LIBNAME option. The URI= option is the recommended way to connect to a Spark data source in SAS 9.4M8.

---

Information about using SAS/ACCESS Interface to Spark with Hadoop distributions that support the Hadoop File System (HDFS) is provided in the ["LIBNAME Statement for the Hadoop Engine"](#). This includes information about Read and Write security and related HDFS permissions.

Here is the LIBNAME statement syntax for accessing Spark.

**LIBNAME** *libref* **spark** <connection-options> <LIBNAME-options>

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see ["LIBNAME Statement" in SAS Global Statements: Reference](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

*spark*

specifies the SAS/ACCESS engine name for the Spark interface.

*connection-options*

provides options that are required to connect to Spark SQL.

All connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

**CLASSPATH="JAR-path"**

specifies the path to the JDBC JAR files. The value that you specify for CLASSPATH= overrides the value that is set for the SAS\_ACCESS\_CLASSPATH environment variable.

Default: SAS\_ACCESS\_CLASSPATH environment variable value

Requirement: You must set a value for the CLASSPATH= connection option or the SAS\_ACCESS\_CLASSPATH environment variable.

---

**Note:** Support for this option was added in SAS 9.4M8.

---

**DRIVERCLASS=<">class<">**

specifies the class name of the JDBC driver to use for your connection.

Default: none

Alias: DRIVER=

---

**Note:** The DRIVERCLASS= option is optional but can be specified in cases where the driver does not load automatically or there is ambiguity in which JDBC driver might be used. Typically, this includes drivers that use JDBC 4.0 or earlier.

---

Examples:

- Databricks JDBC driver (available from Databricks):

```
driverclass='com.simba.spark.jdbc.Driver'
```

- Apache Hive JDBC driver:

```
driverclass='org.apache.hive.jdbc.HiveDriver'
```

**HDFS\_DATADIR='path'**

when option HDFS\_METADIR= is specified, specifies the path to the Spark directory where SAS reads and writes data (for example, '/sas/hpa').

---

**Note:** This option is valid only when SAS is configured with access to HDFS.

---

**HDFS\_METADIR='path'**

specifies the path to an HDFS directory that contains XML-based table definitions. Through these descriptors, SAS then accesses the data using HDFS instead of Spark. If you want the Spark engine to connect using Spark SQL, do not specify this option.

---

**Note:** This option is valid only when SAS is configured with access to HDFS.

---

**HDFS\_TEMPDIR='path'**

specifies the path to the HDFS directory where SAS reads and writes temporary data.

---

**Note:** This option is valid only when SAS is configured with access to HDFS.

---

Default: HDFS\_TEMPDIR=' /tmp '

**PASSWORD=<'>Spark-password<'>**

specifies the Spark password that is associated with your user ID. If the password contains spaces or nonalphanumeric characters, you must enclose the password in quotation marks. If you do not want to enter your Spark password in uncoded text on this statement, see PROC PWENCODE in the [Base SAS Procedures Guide](#) for a method to encode it.

Alias: PASS=, PWD=, PW=

---

**Note:** The USER= and PASSWORD= options remain valid. However, these options are less secure because you are saving authentication information in your programs. Therefore, as a better security practice, it is recommended that you use the AUTHDOMAIN= option instead.

---

**SCHEMA=Spark-schema**

specifies the Spark schema to use for your database connection.

Alias: DATABASE=, DB=

Default: default

**URI='jdbc:driver-name://driver-connection-options'**

specifies the JDBC URL to connect to Spark SQL.

See the [Databricks](#) documentation for information about how to construct a Databricks JDBC driver connection string.

See the documentation for your JDBC driver for information to construct a connection string for other Spark data sources.

See “[Spark LIBNAME Statement Examples](#)” for examples of URL values.

Alias: URL=

Default: none

---

**Note:** This option is the recommended way to connect to a Spark data source in [SAS 9.4M8](#).

---

**USER=<'>Spark-user-name<'>**

specifies the user name for Read (JDBC) and Write (HDFS) operations. Do not use the USER= argument if your Hadoop cluster is secured by Kerberos.

Alias: UID=

---

**Note:** USER= must be specified as a connection option and cannot be specified by using the URL= option.

---



---

**Note:** The USER= and PASSWORD= options remain valid. However, these options are less secure because you are saving authentication information in your programs. Therefore, as a better security practice, it is recommended that you use the AUTHDOMAIN= option instead.

---

***LIBNAME options***

specify how SAS processes DBMS objects. The following table describes the LIBNAME options for SAS/ACCESS Interface to Spark, with the applicable default values. This table also identifies LIBNAME options that are valid in the

CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 38.1** SAS/ACCESS LIBNAME Options for Spark

| Option                   | Default Value | Valid in CONNECT |
|--------------------------|---------------|------------------|
| ACCESS=                  | none          |                  |
| ANALYZE=                 | NO            |                  |
| AUTHDOMAIN=              | none          |                  |
| BL_PORT=                 | 8020          |                  |
| BULKLOAD=                | YES           |                  |
| CONNECTION=              | SHAREDREAD    | •                |
| DBCONINIT=               | none          | •                |
| DBCONTERM=               | none          | •                |
| DBCREATE_TABLE_EXTERNAL= | NO            | •                |
| DBCREATE_TABLE_OPTS=     | none          | •                |
| DBGEN_NAME=              | DBMS          |                  |
| DBLIBINIT=               | none          |                  |
| DBLIBTERM=               | none          |                  |
| DBMAX_TEXT=              | 1024          | •                |
| DBMSTEMP=                | NO            |                  |
| DBSASLABEL=              | COMPAT        |                  |
| DEFER=                   | NO            | •                |
| DIRECT_SQL=              | YES           |                  |
| IGNORE_BASELINE=         | NO            | •                |
| LOGIN_TIMEOUT=           | 30            | •                |
| MULTI_DATASRC_OPT=       | NONE          |                  |
| POST_STMT_OPTS=          | none          |                  |

| Option              | Default Value                                | Valid in CONNECT |
|---------------------|----------------------------------------------|------------------|
| PRESERVE_COL_NAMES= | none                                         |                  |
| PRESERVE_TAB_NAMES= | NO                                           |                  |
| PROPERTIES=         | none                                         | •                |
| QUERY_TIMEOUT=      | 0                                            | •                |
| READ_METHOD=        | none                                         | •                |
| READBUFF=           | automatically calculated based on row length | •                |
| SCHEMA=             | the Spark schema <code>default</code>        |                  |
| SCRATCH_DB=         | none                                         | •                |
| SPOOL=              | YES                                          |                  |
| SQL_FUNCTIONS=      | none                                         | •                |
| SQL_FUNCTIONS_COPY= | none                                         |                  |
| SUB_CHAR=           | none                                         |                  |
| TRANSCODE_FAIL=     | ERROR                                        | •                |

## Connection Options for the Apache Hive JDBC Driver

When the Apache Hive JDBC driver, `org.apache.hive.jdbc.HiveDriver`, is used, SAS/ACCESS to Spark constructs the JDBC URL based on the contents of the `hive-site.xml` file. This usage is mostly deprecated; however, the following connection options are still supported when using the Apache Hive JDBC driver.

`PORT=port`

specifies the port number to use to connect to the specified Spark service.

Alias: `SERVICE=`

Default: 10016

`PROPERTIES='string';`

specifies additional JDBC connection properties.

Example:

```
properties="hive.mapred.mode=strict"
```

**Note:** This option is ignored when the URL= option is specified.

**SERVER=<'>Spark-server-name<'>**

specifies the Spark server name that runs the Hive service. If the server name contains spaces or nonalphanumeric characters, you must enclose the server name in quotation marks.

Alias: HOST=

## Spark LIBNAME Statement Examples

### Connecting to Databricks in Microsoft Azure

This example accesses Spark data in Databricks on Microsoft Azure. The Databricks JDBC driver is recommended for accessing Spark data in Databricks.

```
libname spkdbk spark classpath='/JAR-path'
  driverclass="com.simba.spark.jdbc.Driver"
  bulkload=no
  url=' jdbc:spark://server:port/schema;transportMode=http;ssl=1;HTTPPath=myHttpPath;
AuthMech=3;defaultStringColumnLength=255;useNativeQuery=1;<options>'
  user=token
  password=mytoken
;
```

**Note:** When using the Databricks JDBC driver:

- BULKLOAD=NO is required by the cloud host provider.
- Specify useNativeQuery=1 in the JDBC URL. That is because SAS generates native Spark SQL.
- Specify the defaultStringColumnLength= option in the JDBC URL. This option is used to limit the length of STRING columns that are read into SAS.

### Connecting to Apache Spark with HDFS

This example shows the options that are needed to connect to Spark on a Hadoop Distribution that has access to HDFS. Consult the documentation for your JDBC driver to create the JDBC URL that is specified in the URI= option. When HDFS is available, SAS/ACCESS Interface to Spark can use HDFS for bulk loading. BULKLOAD=YES is enabled by default.

The SAS\_HADOOP\_CONFIG\_PATH= and SAS\_HADOOP\_JAR\_PATH= environment variables must be set to enable access to HDFS from SAS.

```
option set = SAS_HADOOP_JAR_PATH="Hadoop-JAR-location";
option set= SAS_HADOOP_CONFIG_PATH="Hadoop-configuration-file-location";

libname spkhdbs spark classpath='/JAR-path' driverclass='class-name'
uri='jdbc:driver-name://driver-connection-options'
user=myuserid
password=myuserpwd
;
```

**Note:** Specify QueryPassthrough=True (or the driver equivalent) in the JDBC URL, because SAS generates native Spark SQL. Consider setting an option that controls the column length as well.

## Data Set Options for Spark

All SAS/ACCESS data set options in this table are supported for Spark. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

**Table 38.2** SAS/ACCESS Data Set Options for Spark

| Option                   | Default Value                                    |
|--------------------------|--------------------------------------------------|
| ANALYZE=                 | LIBNAME option value                             |
| BULKLOAD=                | YES                                              |
| COLUMN_DELIMITER=        | \001 (Ctrl-A)                                    |
| DBCONDITION=             | none                                             |
| DBCREATE_TABLE_EXTERNAL= | NO                                               |
| DBCREATE_TABLE_LOCATION= | depends on how your Hadoop cluster is configured |
| DBCREATE_TABLE_OPTS=     | LIBNAME option value                             |
| DBFORCE=                 | NO                                               |
| DBGEN_NAME=              | DBMS                                             |
| DBLARGETABLE=            | none                                             |

| Option           | Default Value                                         |
|------------------|-------------------------------------------------------|
| DBMAX_TEXT=      | 1024                                                  |
| DBSASLABEL=      | COMPAT                                                |
| DBSASTYPE=       | see <a href="#">Data Types for Spark on page 1312</a> |
| DBTYPE=          | see <a href="#">Data Types for Spark on page 1312</a> |
| POST_STMT_OPTS=  | none                                                  |
| POST_TABLE_OPTS= | none                                                  |
| PRE_STMT_OPTS=   | none                                                  |
| PRE_TABLE_OPTS=  | none                                                  |
| QUERY_TIMEOUT=   | LIBNAME option value                                  |
| READ_METHOD=     | none                                                  |
| READBUFF=        | LIBNAME option value                                  |
| SCHEMA=          | LIBNAME option value                                  |
| SCRATCH_DB=      | none                                                  |
| SUB_CHAR=        | none                                                  |
| TRANSCODE_FAIL=  | ERROR                                                 |

---

## SQL Pass-Through Facility Specifics for Spark

---

### Key Information

For general information about this feature, see “[Overview of SQL Procedure Interactions with SAS/ACCESS](#)” on page 689.

Here are the SQL pass-through facility specifics for the Spark interface.

- The *dbms-name* is SPARK.

- The CONNECT statement is required.
- PROC SQL supports multiple connections to Spark. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default SPARK alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).

## CONNECT Statement Examples

### Connect Using an Already Assigned Libref

When a Spark libref is already assigned, you can use PROC SQL's CONNECT USING syntax to avoid re-typing the connection options.

```
/* libref x is already assigned */
proc sql;
  connect using x;
  /* get the Spark SQL version */
  select * from connection to x (select version () );
  /* create a table using Spark SQL, not SAS SQL */
  execute (create table if not exists test1 (i integer) ) by x;
  disconnect from x;
quit;
```

### Connect Using Connection Parameters

When a Spark libref is not available, you can connect to Spark SQL using PROC SQL's CONNECT TO syntax. You must provide the same connection parameters that are required in the LIBNAME statement in the CONNECT TO statement.

This example uses the Databricks JDBC driver. The database connection parameters are assigned to the alias y.

```
proc sql;
  connect to spark as y (
    driverClass="com.simba.spark.jdbc.Driver"
    bulkload=no
    url='jdbc:spark://server:port//schema;transportMode=http;ssl=1;
    HTTPPath=myHttpPath;AuthMech=3;defaultStringColumnLength=255;useNativeQuery=1'
    user=token
    password=mytoken
  );
  /* get the Spark SQL version */
  select * from connection to y (select version () );
  /* create a table using Spark SQL, not SAS SQL */
```

```

execute (create table if not exists test1 (i integer) ) by y;

disconnect from y;
quit;

```

## Passing SAS Functions to Spark

SAS/ACCESS Interface to Spark passes the following SAS functions to Spark for processing. Where the Spark function name differs from the SAS function name, the Spark name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                           |                   |
|---------------------------|-------------------|
| ** (POWER)                | LOG10             |
| ABS                       | LOG2              |
| ARCOS (ACOS)              | LOWCASE (LOWER)   |
| ARSIN (ASIN)              | MINUTE            |
| ATAN                      | MOD (CAST)        |
| CAT (CONCAT)              | MONTH             |
| CEIL                      | QTR (QUARTER)     |
| COMPRESS (REGEXP_REPLACE) | SCAN (SPLIT)      |
| COS                       | SECOND            |
| COUNT                     | SIGN              |
| DAY                       | SIN               |
| DTEXTDAY (DAY)            | SOUNDEX           |
| DTEXTMONTH (MONTH)        | SQRT              |
| DTEXTWEEKDAY              | STD (STDDEV_SAMP) |
| DTEXTYEAR (YEAR)          | STRIP (TRIM)      |
| EXP                       | SUBSTR            |
| FLOOR                     | TAN               |
| HOUR                      | TRIMN (RTRIM)     |
| INDEX (LOCATE)            | UPCASE (UPPER)    |
| LEFT (LTRIM)              | VAR (VAR_SAMP)    |
| LENGTH (LENGTH(RTRIM))    | YEAR              |
| LOG (LN)                  |                   |

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding Spark functions that are passed down to Hadoop. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to Spark. Due to incompatibility in date and time functions between Spark and SAS, Spark might not process them correctly. Check your results to determine whether these functions are working as expected.

```

COALESCE
DATE (TO_DATE(FROM_UNIX_TIMESTAMP))
DATEPART (TO_DATE(FROM_UNIX_TIMESTAMP))
DATETIME (FROM_UNIXTIME(UNIX_TIMESTAMP))

```

```

REPEAT
ROUND
TODAY (TO_DATE(FROM_UNIXTIME(UNIX_TIMESTAMP)))
TRANSTRN (REGEXP_REPLACE)

```

---

## Passing Joins to Spark

For a multiple libref join to pass to Spark, all of these components of the LIBNAME statements must match exactly.

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- port (PORT=)
- schema (SCHEMA=, where the URI= host or port should match if you specified URI=)

You can use the [SQL pass-through facility](#) to pass a cross-schema join to Spark. For more information, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Bulk Loading for Spark

---

### Overview

Bulk loading of data is supported for Spark data sources that have access to the Hadoop Distributed File System (HDFS). Bulk loading is not available in Databricks.

Your installation has HDFS access configured if you can use PROC HADOOP. For information to configure access to HDFS, see the Spark information in [SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS](#). Also see “[Connecting to Apache Spark with HDFS](#)”.

---

### Loading

When HDFS access is configured, SAS/ACCESS Interface to Spark makes no differentiation between bulk loading and a standard load process. As a result, the [BULKLOAD=YES](#) syntax is supported, but it is not required.

Here is how a text-based table (STORED AS TEXTFILE) is created.

- 1 SAS issues a CREATE TABLE Spark command to the Spark server. The command contains all table metadata (column definitions) and the table properties that are specific to SAS that refine Spark metadata to handle maximum string lengths and date/time formats.
- 2 SAS uses HDFS to upload table data to the HDFS /tmp directory. The resulting file is a UTF-8-delimited text file that by default uses CTRL-A ('\001') as a field delimiter and newline ('\n') as a record separator.
- 3 SAS issues a LOAD DATA command to move the data file from the HDFS /tmp directory to the appropriate Spark warehouse location. The data file is now part of the Spark table.

Here is how a non-text table is created:

- 1 Specify the DBCREATE\_TABLE\_OPTS= data set option containing a Spark STORED AS clause to the new table reference. Here is an example:

```
data spk.new_hive_table(DBCREATE_TABLE_OPTS='STORED AS SEQUENCEFILE');
  set sas_table;
  run;
```

- 2 SAS issues two CREATE TABLE statements to the Spark server. One CREATE TABLE statement creates the target Spark table. The other CREATE TABLE statement creates a temporary table.
- 3 SAS uses HDFS to upload table data to the HDFS /tmp directory. The resulting file is a UTF-8-delimited text file.
- 4 SAS issues a LOAD DATA command to move the data file from the HDFS /tmp directory into the temporary table.
- 5 SAS issues an INSERT INTO statement that copies and transforms the temp table text data into the target (non-text) Spark table.
- 6 SAS deletes the temporary table.

Spark considers a table to be a collection of files in a directory that bears the table name. The CREATE TABLE command creates this directory either directly in the Spark warehouse or in a subdirectory, if a nondefault schema is used. The LOAD DATA command moves the data to the correct location.

When PROC APPEND is used to append to the Spark table, the Spark interface places data in a new HDFS file. The interface then issues either the LOAD DATA pattern or the LOAD DATA plus INSERT INTO pattern described earlier.

## Examples

This example creates and loads the FLIGHTS98 HiveServer2 table from the SASFLT.FLT98 SAS data set.

```
libname sasflt 'SAS-library';
libname spk_air spark <connection-options>

proc sql;
create table spk_air.flights98
  as select * from sasflt.flt98;
quit;
```

This example creates and loads the ALLFLIGHTS HiveServer2 table in SEQUENCEFILE format from the SASFLT.ALLFLIGHTS SAS data set.

```
data spk_air.allflights (dbcreate_table_opts='stored as sequencefile');
  set sasflt.allflights;
run;
```

In this example, the SASFLT.FLT98 SAS data set is appended to the ALLFLIGHTS HiveServer2 table.

```
proc append base=spk_air.allflights data=sasflt.flt98;
run;
```

---

## Naming Conventions for SAS and Spark

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names”](#).

Current versions of Spark are case insensitive for comparisons, but case is preserved for display purposes. By default, SAS converts them to uppercase. Users can set the PRESERVE\_COL\_NAMES= and PRESERVE\_TAB\_NAMES= options (shortcut PRESERVE\_NAMES=) to preserve the case of identifiers. Doing this usually is not required unless the case must be preserved for display purposes.

Spark does not currently permit a table name to begin with the underscore (\_) character.

- Spark currently does not permit a period (.) or colon (:) within a column name, and a table name cannot begin with the underscore (\_) character.
- A SAS name must be from 1 to 32 characters long. When Spark column names and table names are 32 characters or less, SAS handles them seamlessly. When SAS reads Spark column names that are longer than 32 characters, a generated SAS variable name is truncated to 32 characters. Spark table names should be 32 characters or less because SAS cannot truncate a table reference. If you already have a table name that is greater than 32 characters, create a Spark table view or use the explicit SQL feature of PROC SQL to access the table.
- If truncating would result in identical names, SAS generates a unique name.
- Further naming restrictions might apply based on the Spark distribution. For example, some older Spark distributions do not support UTF-8 column names.

---

# Data Types for Spark

---

## Supported Spark Data Types

Here are the Spark data types that the Spark engine supports.

- Numeric data

|         |          |
|---------|----------|
| BIGINT  | FLOAT    |
| BOOLEAN | INT      |
| DECIMAL | SMALLINT |
| DOUBLE  | TINYINT  |

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

---

- Character data

|          |             |
|----------|-------------|
| BINARY   | STRING      |
| CHAR $n$ | VARCHAR $n$ |

- Date and time data

|          |           |
|----------|-----------|
| DATE     | TIMESTAMP |
| INTERVAL |           |

- Complex data

|       |        |
|-------|--------|
| ARRAY | STRUCT |
| MAP   |        |

---

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Spark assigns to SAS variables when using the [LIBNAME statement on page 1299](#) to read from a Spark table. These default formats are based on Spark column attributes.

**Table 38.3 LIBNAME Statement: Default SAS Formats for Spark Data Types**

| <b>Spark Data Type</b>             | <b>SAS Data Type</b> | <b>Default SAS Format</b> |
|------------------------------------|----------------------|---------------------------|
| ARRAY                              | character            | none                      |
| BINARY                             | character            | \$HEX32767                |
| BOOLEAN                            | numeric              | w. (1.)                   |
| CHAR( <i>n</i> ) <sup>1</sup>      | character            | \$ <i>w</i> .             |
| DATE                               | numeric              | DATE9.                    |
| DECIMAL( <i>p,s</i> ) <sup>2</sup> | numeric              | <i>w.d</i>                |
| DOUBLE                             | numeric              | <i>w</i> .                |
| FLOAT                              | numeric              | <i>w</i> .                |
| INT                                | numeric              | w. (11.)                  |
| INTERVAL                           | numeric              | DATETIME25.6              |
| MAP                                | character            | none                      |
| SMALLINT                           | numeric              | w. (6.)                   |
| STRING <sup>3</sup>                | character            | \$32767.                  |
| STRUCT                             | character            | none                      |
| TIMESTAMP                          | numeric              | DATETIME25.6              |
| TINYINT                            | numeric              | w. (4.)                   |
| VARCHAR( <i>n</i> ) <sup>1</sup>   | character            | \$ <i>w</i> .             |

<sup>1</sup> *n* in Spark data types is equivalent to *w*. in SAS formats.<sup>2</sup> *p,s* in Spark data types is equivalent to *w.d* in SAS formats<sup>3</sup> The STRING data type can be mapped to the VARCHAR data type by the JDBC client driver. For more information, check the JDBC vendor's documentation.

This table shows the default Spark data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

**Table 38.4 LIBNAME Statement: Default Spark Data Types for SAS Variable Formats**

| <b>SAS Variable Format</b> | <b>Spark Data Type</b> |
|----------------------------|------------------------|
| <i>w.d</i>                 | DOUBLE                 |

| SAS Variable Format       | Spark Data Type                |
|---------------------------|--------------------------------|
| w.                        | INT, SMALLINT, TINYINT, BIGINT |
| \$w.                      | VARCHAR                        |
| datetime formats          | TIMESTAMP                      |
| date formats              | DATE                           |
| time formats <sup>1</sup> | VARCHAR                        |

<sup>1</sup> A column created in Spark using a TIME format will be created as a VARCHAR column with a SASFMT format.

## Spark Null Values

Spark has a special value called NULL. A Spark NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Spark NULL value, it interprets it as a SAS missing value. For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how the DBMS handles SAS missing character values, use the NULLCHAR= and NULLCHARVAL= data set options.

## SAS Table Properties for Hive and Spark

When creating a table in HiveQL or Spark SQL with STRING columns, SAS might add extended type information to the table. The extended type information is added by including a TBLPROPERTIES clause on the CREATE TABLE query with SASFMT key-value pairs. This feature saves the data type information, as it is known to SAS, in the foreign table’s metadata. SASFMT table properties were originally developed for older versions of HiveQL and Spark SQL that did not support basic SQL data types such as CHAR, DATE, TIMESTAMP, and VARCHAR. As current versions of HiveQL and Spark SQL have come to support these data types, it is no longer necessary to specify SASFMT table properties, although they are still supported for backward compatibility. The CHAR, DATE, TIMESTAMP, and VARCHAR Hive and Spark types are used instead.

There are two cases where the SASFMT table properties can still be used.

- HiveQL and Spark SQL do not support the TIME data type. Therefore, SAS continues to use the SASFMT table properties when creating a table with the TIME data type. The SAS TIME column is saved as a VARCHAR column with its associated SASFMT table property. Here is an example of the TBLPROPERTIES that are generated by SAS.

```
libname x spark <connection-options>;
proc sql;
```

```

connect using x;
execute (drop table m1test) by x;
quit;

proc sql;
  create table x.m1test(dbtype =(v=string)) (v char(10));
quit;

```

Here is the output in the log output:

```

CREATE TABLE `accesstesting`.`TIMETEST` (`t` VARCHAR(20)) TBLPROPERTIES ('SAS
OS Name='Linux', 'SAS Version='version-number', 'SASFMT:t'='TIME(8.0)')

```

- Advanced users can add SASFMT table properties to an existing table to refine the data type or length as it is known to SAS by using explicit SQL pass-through to issue a HiveQL or Spark SQL ALTER TABLE query. In the example below, a table is created with the STRING data type using explicit SQL. Explicit SQL is used again to add the SASFMT table property. The PROC SQL DESCRIBE TABLE statement describes the table as it is known to SAS.

```

proc sql;
  connect using x;
  execute (create table stringtest (s string)) by x;
  execute (alter table stringtest SET TBLPROPERTIES ('SASFMT:s'='TIME(8.0)')) by x;
quit;

proc sql;
  describe table x.stringtest;
quit;

```

Here is the output from the DESCRIBE TABLE statement:

```

NOTE: SQL table X.STRINGTEST was created like:
create table X.STRINGTEST (
  s num format=TIME8. informat=TIME8. label='s'
);

```

# Sample Programs for Spark

## Code Snippets

The code snippets in this section resemble those for most other SAS/ACCESS interfaces.

This snippet shows a list of available Spark tables.

```
proc datasets lib=spk; quit;
```

Here is the metadata for the mytab Spark table.

```
proc contents data=spk.mytab; quit;
```

This snippet extracts mytab data into SAS.

```
data work.a;
set spk.mytab;
run;
```

This extracts a subset of the mytab rows and columns into SAS. Subsetting the rows (with a WHERE statement, for example) can help avoid extracting too much data into SAS.

```
data work.a;
set spk.mytab (keep=col1 col2);
where col2=10;
run;
```

## Use DBSASTYPE= to Load Spark Data into SAS

This example uses the [DBSASTYPE= data set option](#) to load Spark textual dates, timestamps, and times into the corresponding SAS DATE, DATETIME, and TIME formats. The first step reads in a SAS character string to display the data and make clear what occurs in successive steps.

```
data;
  set spk.testSparkDate;
  put dt;
run;
```

```
2011-10-17
2009-07-30 12:58:59
11:30:01
```

```
data;
  set spk.testSparkDate(dbsastype=(dt='date'));
  put dt;
run;
```

```
17OCT2011
30JUL2009
.
```

```
data;
  set spk.testSparkDate(dbsastype=(dt='datetime'));
  put dt;
run;
```

```
17OCT2011:00:00:00
30JUL2009:12:58:59
.
```

```
data;
  set spk.testSparkDate(dbsastype=(dt='time'));
  put dt;
run;
```

```

.
12:58:59
11:30:01

```

This code uses SAS SQL to access a Spark table.

```

proc sql;
create table work.a as select * from spk.newtab;
quit;

```

SAS data is then loaded into Spark.

```

data spk.newtab2;
set work.a;
run;

```

Use implicit pass-through SQL to extract only 10 rows from the Newtab table and load the work SAS data set with the results.

```

proc sql;
connect to spark (<connection-options>);
create table work.a as
  select * from connection to spark (select * from newtab limit 10);
quit;

```

## Create a Partitioned Table with a File Type of SEQUENCEFILE

Use the DBCREATE\_TABLE\_OPTS value PARTITIONED BY (*column data-type*) STORED AS SEQUENCEFILE.

```

libname spk SPARK <connection-options>;

data spk.part_tab (DBCREATE_TABLE_OPTS="PARTITIONED BY (s2 int, s3
string)
STORED AS SEQUENCEFILE");
set work.part_tab;
run;

```

You can view additional sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.



# SAS/ACCESS Interface to Teradata

---

|                                                                        |             |
|------------------------------------------------------------------------|-------------|
| <b><i>System Requirements for SAS/ACCESS Interface to Teradata</i></b> | <b>1321</b> |
| <b><i>Introduction to SAS/ACCESS Interface to Teradata</i></b>         | <b>1321</b> |
| Overview                                                               | 1321        |
| The SAS/ACCESS Teradata Client                                         | 1322        |
| Restrictions for the ACCESS and DBLOAD Procedures                      | 1322        |
| <b><i>LIBNAME Statement for the Teradata Engine</i></b>                | <b>1323</b> |
| Overview                                                               | 1323        |
| Arguments                                                              | 1323        |
| <b><i>Data Set Options for Teradata</i></b>                            | <b>1328</b> |
| <b><i>SQL Pass-Through Facility Specifics for Teradata</i></b>         | <b>1332</b> |
| Key Information                                                        | 1332        |
| Including Comments in SQL Code                                         | 1333        |
| SQL Examples                                                           | 1334        |
| <b><i>Temporary Table Support for Teradata</i></b>                     | <b>1335</b> |
| Overview                                                               | 1335        |
| About Volatile Temporary Tables in Teradata                            | 1335        |
| <b><i>Passing SAS Functions to Teradata</i></b>                        | <b>1336</b> |
| <b><i>Passing Joins to Teradata</i></b>                                | <b>1337</b> |
| <b><i>Maximizing Teradata Load and Read Performance</i></b>            | <b>1337</b> |
| Overview                                                               | 1337        |
| Using the TPT API                                                      | 1338        |
| Legacy Load and Read Utilities                                         | 1344        |
| Using FastLoad                                                         | 1344        |
| Using MultiLoad                                                        | 1346        |
| MultiLoad Examples                                                     | 1348        |
| <b><i>Autopartitioning Scheme for Teradata (Legacy)</i></b>            | <b>1350</b> |
| Overview                                                               | 1350        |
| FastExport and Case Sensitivity                                        | 1350        |
| FastExport Password Security                                           | 1351        |
| FastExport Setup                                                       | 1351        |

|                                                                          |             |
|--------------------------------------------------------------------------|-------------|
| Using FastExport .....                                                   | 1352        |
| FastExport and Explicit SQL .....                                        | 1352        |
| Exceptions to Using FastExport .....                                     | 1353        |
| Threaded Reads with Partitioning WHERE Clauses .....                     | 1353        |
| FastExport versus Partitioning WHERE Clauses .....                       | 1354        |
| <b>Teradata Processing Tips for SAS Users .....</b>                      | <b>1354</b> |
| Reading from and Inserting to the Same Teradata Table .....              | 1354        |
| Using a BY Clause to Order Query Results .....                           | 1355        |
| Using TIME and TIMESTAMP .....                                           | 1356        |
| Replacing PROC SORT with a BY Clause .....                               | 1357        |
| Reducing Workload on Teradata by Sampling .....                          | 1358        |
| <b>Security: Authentication with Teradata .....</b>                      | <b>1358</b> |
| Authentication Using TD2 .....                                           | 1358        |
| Authentication Using Kerberos .....                                      | 1359        |
| Authentication Using LDAP .....                                          | 1360        |
| Using the Teradata Wallet Feature .....                                  | 1361        |
| <b>Locking in the Teradata Interface .....</b>                           | <b>1363</b> |
| Overview .....                                                           | 1363        |
| Understanding SAS/ACCESS Locking Options .....                           | 1364        |
| When to Use SAS/ACCESS Locking Options .....                             | 1365        |
| When Lock Options Are Not Recognized .....                               | 1366        |
| Examples .....                                                           | 1367        |
| <b>Naming Conventions for Teradata .....</b>                             | <b>1369</b> |
| Teradata Conventions .....                                               | 1369        |
| SAS Naming Conventions .....                                             | 1370        |
| Naming Objects to Meet Teradata and SAS Conventions .....                | 1370        |
| Accessing Teradata Objects That Do Not Meet SAS Naming Conventions ..... | 1370        |
| <b>Data Types for Teradata .....</b>                                     | <b>1371</b> |
| Supported Teradata Data Types .....                                      | 1371        |
| Teradata Null Values .....                                               | 1372        |
| LIBNAME Statement Data Conversions .....                                 | 1373        |
| Working with NUMBER Data .....                                           | 1375        |
| Data Returned as SAS Binary Data with Default Format \$HEX .....         | 1376        |
| <b>Temporal Data Types for Teradata .....</b>                            | <b>1377</b> |
| Overview .....                                                           | 1377        |
| Supported Temporal Data Types .....                                      | 1378        |
| Specifying Transaction Time and Valid Time .....                         | 1378        |
| Creating a Table from SAS with the PERIOD Data Type .....                | 1378        |
| Reading in a PERIOD Data Type .....                                      | 1379        |
| Temporal Qualifiers .....                                                | 1379        |
| <b>Sample Programs for Teradata .....</b>                                | <b>1380</b> |

---

# System Requirements for SAS/ACCESS Interface to Teradata

You can find information about system requirements for SAS/ACCESS Interface to Teradata in the following locations:

- [System Requirements for SAS/ACCESS Interface to Teradata with SAS 9.4](#)
- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to Teradata

---

## Overview

For available SAS/ACCESS features, see [Teradata supported features on page 118](#). For more information about Teradata, see your Teradata documentation.

---

**Note:** Beginning with SAS 9.4M8, SAS/ACCESS to Teradata on the z/OS platform is no longer available. If you have an instance of SAS/ACCESS Interface to Teradata on z/OS and plan to upgrade to SAS 9.4M8 or later, SAS recommends that you unconfigure and uninstall it. For more information, see [Unconfiguring and Uninstalling Retired Products](#).

---

SAS/ACCESS Interface to Teradata includes SAS Data Connector to Teradata. If you have the appropriate license, you might also have access to the SAS Data Connect Accelerator to Teradata. The data connector or data connect accelerator enables you to load large amounts of data into the CAS server for parallel processing. For more information, see these sections:

- “[Where to Specify Data Connector Options](#)” in *SAS Cloud Analytic Services: User’s Guide*
- “[Teradata Data Connector](#)” in *SAS Cloud Analytic Services: User’s Guide*

---

## The SAS/ACCESS Teradata Client

Teradata is a massively parallel (MPP) RDBMS. A high-end Teradata server supports many users. It simultaneously loads and extracts table data and processes complex queries. Upsert processing is also supported using MultiLoad.

Because Teradata customers run many processors at the same time for queries of the database, users enjoy excellent DBMS *server* performance. The challenge to client software, such as SAS, is to leverage Teradata performance by rapidly extracting and loading table data. SAS/ACCESS Interface to Teradata meets this challenge by letting you optimize extracts and loads (reads and creates).

Information throughout this document explains how you can use the SAS/ACCESS interface to optimize DBMS operations.

- It supports the Teradata Parallel Transporter (TPT) API on Windows and UNIX. This API uses the Teradata Load, Update, and Stream operators to load data and the export operator to read data.
- It can create and update Teradata tables. It supports a FastLoad interface that rapidly creates new table. It can also potentially optimize table reads by using FastExport for the highest possible read performance.
- It supports MultiLoad, which loads both empty and existing Teradata tables and greatly accelerates the speed of insertion into Teradata tables.

---

## Restrictions for the ACCESS and DBLOAD Procedures

SAS/ACCESS Interface to Teradata does not support the ACCESS and DBLOAD procedures. The LIBNAME engine technology enhances and replaces the functionality of these procedures. Therefore, you must revise SAS jobs that were written for a different SAS/ACCESS interface and that include ACCESS or DBLOAD procedures. You need to do this revision before you can run the SAS jobs with SAS/ACCESS Interface to Teradata.

---

# LIBNAME Statement for the Teradata Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Teradata supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Teradata.

**LIBNAME** *libref* **teradata** <connection-options> <LIBNAME-options>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

### *libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

### *teradata*

specifies the SAS/ACCESS engine name for the Teradata interface.

### *connection-options*

provides connection information and controls how SAS manages the timing and concurrence of the connection to the DBMS. Here are the connection options for the Teradata interface.

---

**Note:** All of these connection options, except SCHEMA=, are valid when used in the CONNECT statement with the SQL procedure.

---

**USER=<'>*Teradata-user-name*<'> | <">*Idapid@LDAP*<"> | <">*Idapid@LDAPrealm-name*<">**

specifies a required connection option that specifies a Teradata user name. If the name contains blanks or national characters, enclose it in quotation marks.

For more information about LDAP, see “[Authentication Using LDAP](#)” on page 1360.

**PASSWORD=<'>*Teradata-password*<'>**

is a required connection option that specifies a Teradata password. The password that you specify must be correct for your USER= value. If the password contains spaces or nonalphanumeric characters, you must enclose

it in quotation marks. If you do not want to enter your Teradata password in clear text on this statement, see PROC PWENCODE in the [Base SAS Procedures Guide](#) for a method for encoding it. For LDAP authentication, use this password option to specify the authentication string or password.

**ACCOUNT=<'>account\_ID<'>**

is an optional connection option that specifies the account number that you want to charge for the Teradata session.

**TDPID=<'>dbname<'>**

Alias: SERVER=

specifies a required connection option if you run more than one Teradata server. TDPID= operates differently for network-attached and channel-attached systems, as described below.

- For NETWORK-ATTACHED systems (PC and UNIX), *dbname* specifies an entry in your (client) HOSTS file that provides an IP address for a database server connection.

By default, SAS/ACCESS connects to the Teradata server that corresponds to the *dbccop1* entry in your HOSTS file. When you run only one Teradata server and your HOSTS file specifies the *dbccop1* entry correctly, you do not need to specify TDPID=.

However, if you run more than one Teradata server, you must use the TDPID= option to specifying a *dbname* of eight characters or less.

SAS/ACCESS adds the specified *dbname* to the logon string that it submits to Teradata. (Teradata documentation refers to this name as the *tdpid* component of the logon string.)

After SAS/ACCESS submits a *dbname* to Teradata, Teradata searches your HOSTS file for all entries that begin with the same *dbname*. For Teradata to recognize the HOSTS file entry, the *dbname* suffix must be COPx (x is a number). If there is only one entry that matches the *dbname*, x must be 1. If there are multiple entries for the *dbname*, x must begin with 1 and increment sequentially for each related entry. (See the example HOSTS file entries below.)

When there are multiple, matching entries for a *dbname* in your HOSTS file, Teradata does simple load balancing by selecting one of the Teradata servers that are specified for logon. Teradata distributes your queries across these servers so that it can return your results as fast as possible.

The TDPID= examples below assume that your HOSTS file contains these *dbname* entries and IP addresses.

- Example 1: The TDPID= option is not specified, establishing a logon to the Teradata server that runs at 10.25.20.34: dbccop1 10.25.20.34
- Example 2: Using TDPID= myserver or SERVER=myserver, you specify a logon to the Teradata server that runs at 130.96.8.207: myservercop1 130.96.8.207
- Example 3: Using TDPID=xyz or SERVER=xyz, you specify a logon to a Teradata server that runs at 11.22.33.44 or to a Teradata server that runs at 33.44.55.66: xyzcop1 33.44.55.66 or xyzcop2 11.22.33.44

To support data parcels that are longer than 64K, the Teradata interface queries the Teradata database to assess Alternate Parcel Header (APH) extended-parcel sizes. In such cases, it needs the Teradata server name (*dbname*) as an entry in the HOSTS file or NAMES database to successfully configure the Teradata client for APH processing. The value of the SERVER=

(TDPID=) LIBNAME option must be the unqualified short name, not the fully qualified IP name (for example, SERVER=foo rather than SERVER=foo.my.unx.com).

- For CHANNEL-ATTACHED systems (z/OS), TDPID= specifies the subsystem name. This name must be TDPx, where x can be 0–9, A–Z (not case sensitive), or \$, # or @. If there is only one Teradata server, and your z/OS system administrator has set up the HSISPB and HSHSPB modules, you do not need to specify TDPID=. For further information, see your Teradata TDPID documentation for z/OS.

**DATABASE=<'>database-name<'>**

Alias: DB=

specifies an optional connection option that establishes a LIBNAME connection to the specified DATABASE= name. The database name that you specify becomes the default database. This option enables the user to directly access another database as the user's default database, eliminating the need to qualify database objects. The user is actually logged in to the specified database. You must first set permissions to allow user access. If you do not specify DATABASE= in the LIBNAME statement, then the user's default database is used. This usually is a database with the same name as the user name.

**SCHEMA=<'>database-name<'>**

specifies an optional value that you can use to fully qualify objects in another database for SQL processing. Using the SCHEMA= option does not establish a physical connection to the specified schema.

#### *LIBNAME options*

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Teradata, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For more information, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 39.1** SAS/ACCESS LIBNAME Options for Teradata

| Option                    | Default Value                                                  | Valid in CONNECT |
|---------------------------|----------------------------------------------------------------|------------------|
| ACCESS=                   | none                                                           |                  |
| AUTHDOMAIN=               | none                                                           | •                |
| BULKLOAD=                 | NO                                                             | •                |
| CAST=                     | none                                                           |                  |
| CAST_OVERHEAD_MAXPERCENT= | 20%                                                            |                  |
| CONNECTION=               | UNIQUE for network attached systems<br>(UNIX and PC platforms) | •                |

| Option               | Default Value                                      | Valid in CONNECT |
|----------------------|----------------------------------------------------|------------------|
|                      | SHAREDREAD for channel-attached systems (z/OS)     |                  |
| CONNECTION_GROUP=    | none                                               | •                |
| DBCLIENT_MAX_BYTES=  | 1                                                  | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows) |                  |
| DBCONINIT=           | none                                               | •                |
| DBCONTERM=           | none                                               | •                |
| DBCREATE_TABLE_OPTS= | none                                               |                  |
| DBGEN_NAME=          | DBMS                                               | •                |
| DBINDEX=             | NO                                                 |                  |
| DBLIBINIT=           | none                                               |                  |
| DBLIBTERM=           | none                                               |                  |
| DBMSTEMP=            | NO                                                 |                  |
| DBPROMPT=            | NO                                                 | •                |
| DBSASLABEL=          | COMPAT                                             |                  |
| DBSLICEPARM=         | THREADED_APPS,2                                    |                  |
| DEFER=               | NO                                                 | •                |
| DIRECT_EXE=          |                                                    |                  |
| DIRECT_SQL=          | YES                                                |                  |
| ERRLIMIT=            | none                                               |                  |
| FASTEXPORT=          | NO                                                 | •                |
| LOGDB=               | Default Teradata database for the libref           | •                |
| MODE=                | ANSI                                               | •                |

| Option                 | Default Value                                                                                            | Valid in CONNECT |
|------------------------|----------------------------------------------------------------------------------------------------------|------------------|
| MULTISTMT=             | NO                                                                                                       | •                |
| MULTI_DATASRC_OPT=     | IN_CLAUSE                                                                                                |                  |
| OVERRIDE_RESP_LEN=     | NO                                                                                                       | •                |
| POST_STMT_OPTS=        | none                                                                                                     |                  |
| PRESERVE_COL_NAMES=    | YES                                                                                                      |                  |
| PRESERVE_TAB_NAMES=    | YES                                                                                                      |                  |
| QUERY_BAND=            | none                                                                                                     | •                |
| READ_ISOLATION_LEVEL=  | see “Locking in the Teradata Interface”                                                                  | •                |
| READ_LOCK_TYPE=        | none                                                                                                     | •                |
| READ_MODE_WAIT=        | none                                                                                                     | •                |
| REREAD_EXPOSURE=       | NO                                                                                                       |                  |
| SAS_DBMS_AUTOMETADATA= | NO                                                                                                       |                  |
| SCHEMA=                | none                                                                                                     |                  |
| SESSIONS=              | none                                                                                                     | •                |
| SLEEP=                 | 6                                                                                                        | •                |
| SPOOL=                 | YES                                                                                                      |                  |
| SQL_FUNCTIONS=         | none                                                                                                     |                  |
| SQL_FUNCTIONS_COPY=    | none                                                                                                     |                  |
| SQLGENERATION=         | none                                                                                                     |                  |
| TEMPORAL_QUALIFIER=    | CURRENT<br>VALIDTIME for valid-time column;<br>CURRENT<br>TRANSACTIONTIME<br>for transaction-time column | •                |

| Option                  | Default Value                                  | Valid in CONNECT |
|-------------------------|------------------------------------------------|------------------|
| TENACITY=               | 0 for FastLoad; 4 for FastExport and MultiLoad | •                |
| TPT=                    | YES                                            | •                |
| TPT_DATA_ENCRYPTION=    | NO                                             | •                |
| TPT_MAX_SESSIONS=       | 4                                              | •                |
| TPT_MIN_SESSIONS=       | 1                                              | •                |
| TPT_UNICODE_PASSTHRU=   | NO                                             | •                |
| TR_ENABLE_INTERRUPT=    | NO                                             | •                |
| UPDATE_ISOLATION_LEVEL= | see “Locking in the Teradata Interface”        | •                |
| UPDATE_LOCK_TYPE=       | none                                           | •                |
| UPDATE_MODE_WAIT=       | none                                           | •                |
| UTILCONN_TRANSIENT=     | NO                                             |                  |

## Data Set Options for Teradata

All SAS/ACCESS data set options in this table are supported for Teradata. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

*Table 39.2 SAS/ACCESS Data Set Options for Teradata*

| Option       | Default Value                                                                                                                                   |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| BL_CONTROL=  | creates a FastExport control file in the temporary file directory that is specified by the UTILLOC= system option with a platform-specific name |
| BL_DATAFILE= | creates a MultiLoad script file in the temporary file directory that is specified by                                                            |

| Option                    | Default Value                                                                                                                                                                                                                  |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | the UTILLOC= system option or with a platform-specific name                                                                                                                                                                    |
| BL_LOG=                   | FastLoad errors are logged in Teradata tables named TableName_SE1_randnum and TableName_SE2_randnum. TableName consists of up to the first 15 characters of the target table name, and randnum is a randomly generated number. |
| BUFFERS=                  | 2                                                                                                                                                                                                                              |
| BULKLOAD=                 | NO                                                                                                                                                                                                                             |
| CAST=                     | none                                                                                                                                                                                                                           |
| CAST_OVERHEAD_MAXPERCENT= | 20%                                                                                                                                                                                                                            |
| DBCOMMIT=                 | LIBNAME option value                                                                                                                                                                                                           |
| DBCONDITION=              | none                                                                                                                                                                                                                           |
| DBCONSTRAINT=             | none                                                                                                                                                                                                                           |
| DBCREATE_TABLE_OPTS=      | LIBNAME option value                                                                                                                                                                                                           |
| DBFORCE=                  | NO                                                                                                                                                                                                                             |
| DBGEN_NAME=               | DBMS                                                                                                                                                                                                                           |
| DBINDEX=                  | LIBNAME option value                                                                                                                                                                                                           |
| DBKEY=                    | none                                                                                                                                                                                                                           |
| DBLABEL=                  | NO                                                                                                                                                                                                                             |
| DBLARGETABLE=             | none                                                                                                                                                                                                                           |
| DBNULL=                   | none                                                                                                                                                                                                                           |
| DBSASLABEL=               | COMPAT                                                                                                                                                                                                                         |
| DBSASTYPE=                | see "Data Types for Teradata"                                                                                                                                                                                                  |
| DBSLICE=                  | none                                                                                                                                                                                                                           |
| DBSLICEPARM=              | THREADED_APPS,2                                                                                                                                                                                                                |
| DBTYPE=                   | see "Data Types for Teradata"                                                                                                                                                                                                  |

| Option                | Default Value        |
|-----------------------|----------------------|
| ERRLIMIT=             | 1                    |
| FASTEXPORT=           | NO                   |
| FASTLOAD=             | NO                   |
| MBUFFSIZE=            | 0                    |
| ML_CHECKPOINT=        | none                 |
| ML_ERROR1=            | none                 |
| ML_ERROR2=            | none                 |
| ML_LOG=               | none                 |
| ML_RESTART=           | none                 |
| ML_WORK=              | none                 |
| MULTILOAD=            | NO                   |
| MULTISTMT=            | NO                   |
| NULLCHAR=             | SAS                  |
| NULLCHARVAL=          | a blank character    |
| POST_STMT_OPTS=       | none                 |
| POST_TABLE_OPTS=      | none                 |
| PRE_STMT_OPTS=        | none                 |
| PRE_TABLE_OPTS=       | none                 |
| PRESERVE_COL_NAMES=   | YES                  |
| QUERY_BAND=           | none                 |
| READ_ISOLATION_LEVEL= | LIBNAME option value |
| READ_LOCK_TYPE=       | LIBNAME option value |
| READ_MODE_WAIT=       | LIBNAME option value |
| SASDATEFMT=           | none                 |

| <b>Option</b>        | <b>Default Value</b>                                                                       |
|----------------------|--------------------------------------------------------------------------------------------|
| SCHEMA=              | LIBNAME option value                                                                       |
| SESSIONS=            | none                                                                                       |
| SET=                 | NO                                                                                         |
| SLEEP=               | 6                                                                                          |
| TEMPORAL_QUALIFIER=  | CURRENT VALIDTIME [valid-time column]<br>CURRENT TRANSACTIONTIME [transaction-time column] |
| TENACITY=            | 0 [FastLoad], 4 [FastExport, MultiLoad]                                                    |
| TPT=                 | YES                                                                                        |
| TPT_APPL_PHASE=      | NO                                                                                         |
| TPT_BLOCK_SIZE=      | 64330 for Teradata 15.10 and lower<br>16775168 for Teradata 16.00 and higher               |
| TPT_BUFFER_SIZE=     | 64                                                                                         |
| TPT_CHECKPOINT_DATA= | none                                                                                       |
| TPT_DATA_ENCRYPTION= | none                                                                                       |
| TPT_ERROR_TABLE_1=   | table_name_ET                                                                              |
| TPT_ERROR_TABLE_2=   | table_name_UV                                                                              |
| TPT_LOG_TABLE=       | table_name_RS                                                                              |
| TPT_MAX_SESSIONS=    | 4                                                                                          |
| TPT_MIN_SESSION=     | 1 session                                                                                  |
| TPT_PACK=            | 20                                                                                         |
| TPT_PACKMAXIMUM=     | NO                                                                                         |
| TPT_RESTART=         | NO                                                                                         |
| TPT_TRACE_LEVEL=     | 1                                                                                          |
| TPT_TRACE_LEVEL_INF= | 1                                                                                          |
| TPT_TRACE_OUTPUT=    | driver name, followed by a timestamp                                                       |

| Option                  | Default Value        |
|-------------------------|----------------------|
| TPT_WORK_TABLE=         | table_name_WT        |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |
| UPDATE_LOCK_TYPE=       | LIBNAME option value |
| UPDATE_MODE_WAIT=       | LIBNAME option value |
| UPSERT=                 | NO                   |
| UPSERT_CONDITION=       | none                 |
| UPSERT_WHERE=           | none                 |

---

## SQL Pass-Through Facility Specifics for Teradata

---

### Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page 690.

Here are the SQL pass-through facility specifics for the Teradata interface.

- The *dbms-name* is TERADATA.
- The CONNECT statement is required.
- The Teradata interface can connect to multiple Teradata servers and to multiple Teradata databases. However, if you use multiple simultaneous connections, you must use an *alias* argument to identify each connection.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).

The MODE= LIBNAME option is available with the CONNECT statement. By default, SAS/ACCESS opens Teradata connections in ANSI mode. In contrast, most Teradata tools, such as BTEQ, run in Teradata mode. If you specify MODE=TERADATA, pass-through connections open in Teradata mode, forcing Teradata mode rules for all SQL requests that are passed to the Teradata DBMS. For example, MODE= impacts transaction behavior and case sensitivity. See your Teradata SQL reference documentation for a complete discussion of ANSI versus Teradata mode.

- By default, SAS/ACCESS opens Teradata in ANSI mode. You must therefore use one of these techniques when you write PROC SQL steps that use the SQL pass-through facility.
  - Specify an explicit COMMIT statement to close a transaction. You must also specify an explicit COMMIT statement after any data definition language (DDL) statement. The examples below demonstrate these rules. For further information about ANSI mode and DDL statements, see your Teradata SQL reference documentation.
  - Specify MODE=TERADATA in your CONNECT statement. When MODE=TERADATA, you do not specify explicit COMMIT statements as described above. When MODE=TERADATA, data processing is not case sensitive. This option is available when you use the LIBNAME statement and also with the SQL pass-through facility.

---

#### **CAUTION**

Do not issue a Teradata DATABASE statement within the EXECUTE statement in PROC SQL. Add the DATABASE= option to your CONNECT statement if you must change the default Teradata database.

---

As a best practice, set the SAS\_NON\_XVIEW\_TABLES environment variable to YES when you invoke SAS. Enabling SAS\_NON\_XVIEW\_TABLES has been shown to improve performance when loading data. Include this code in your SAS command:

```
-set SAS_NON_XVIEW_TABLES YES
```

---

## Including Comments in SQL Code

To use comments in your SQL code, enclose them between ‘/\*’ and ‘\*/’ characters. SAS/ACCESS recognizes that content between these characters is a comment. SAS/ACCESS removes the comment before passing SQL statements to the database.

```
proc sql;
  connect to teradata as teral (user=myuser password=mypass
  server=myserver
  tpt=yes fastexport=yes);
  create table mytable as
  select * from connection to teral
  (select * from classdat /* this is a comment */
   where age > 15 );
  disconnect from teral;
  quit;
```

Although Teradata indicates comments with ‘– –’ characters, if you use ‘– –’ as a comment in PROC SQL, any clauses that follow ‘– –’ are treated as part of the comment.

## SQL Examples

In this example, SAS/ACCESS connects to the Teradata DBMS using the `dbcon` alias.

```
proc sql;
  connect to teradata as dbcon (user=myusr1 pass=mypwd1);
  quit;
```

In the next example, SAS/ACCESS connects to the Teradata DBMS using the `tera` alias, drops and re-creates the `SALARY` table, inserts two rows, and disconnects from the Teradata DBMS. Note that `COMMIT` must follow each DDL statement. `DROP TABLE` and `CREATE TABLE` are DDL statements. The `COMMIT` statement that follows the `INSERT` statement is also required. Otherwise, Teradata rolls back the inserted rows.

```
proc sql;
  connect to teradata as tera ( user=myusr1 password=mypwd1 );
  execute (drop table salary) by tera;
  execute (commit) by tera;
  execute (create table salary (current_salary float, name char(10)))
    by tera;
  execute (commit) by tera;
  execute (insert into salary values (35335.00, 'Dan J.')) by tera;
  execute (insert into salary values (40300.00, 'Irma L.')) by tera;
  execute (commit) by tera;
  disconnect from tera;
  quit;
```

For this example, SAS/ACCESS connects to the Teradata DBMS using the `tera` alias, updates a row, and disconnects from the Teradata DBMS. The `COMMIT` statement causes Teradata to commit the update request. Without the `COMMIT` statement, Teradata rolls back the update.

```
proc sql;
  connect to teradata as tera ( user=myusr1 password=mypwd1 );
  execute (update salary set current_salary=45000
    where (name='Irma L.')) by tera;
  execute (commit) by tera;
  disconnect from tera;
  quit;
```

In this example, SAS/ACCESS uses the `tera2` alias to connect to the Teradata database, selects all rows in the `SALARY` table, displays them using PROC SQL, and disconnects from the Teradata database. No `COMMIT` statement is needed in this example because the operations are reading only data. No changes are made to the database.

```
proc sql;
  connect to teradata as tera2 ( user=myusr1 password=mypwd1 );
  select * from connection to tera2 (select * from salary);
  disconnect from tera2;
  quit;
```

In this next example, MODE=TERADATA is specified to avoid case-insensitive behavior. Because Teradata Mode is used, SQL COMMIT statements are not required.

```
/* Create & populate the table in Teradata mode (case insensitive). */
proc sql;
  connect to teradata (user=myusr1 pass=mypwd1 mode=teradata);
  execute(create table casetest(x char(28)) ) by teradata;
  execute(insert into casetest values('Case Insensitivity Desired')) 
    by teradata;
quit;
/* Query the table in Teradata mode (for case-insensitive match). */
proc sql;
  connect to teradata (user=myusr1 pass=mypwd1 mode=teradata);
  select * from connection to teradata (select * from
  casetest where x='case insensitivity desired');
quit;
```

---

# Temporary Table Support for Teradata

---

## Overview

SAS/ACCESS Interface to Teradata supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

---

## About Volatile Temporary Tables in Teradata

---

Since volatile temporary tables are scoped to a user's session, the best practice is to name volatile temp tables with a unique name to avoid any name collision with permanent tables in the current default database. A volatile temp table name that is concatenated with a sequence number or a timestamp would avoid any collision. For example, suppose you have a user JOE who has a default database of DB1. JOE creates a permanent table FOO1 in database DB1, and then JOE creates a volatile temporary table named FOO1, which is scoped to his user session. This is valid, because although the table name is the same, the tables reside in different storage areas—DB1 and session JOE. However, if JOE references FOO, such as a SELECT \* FROM FOO statement, then there is an error from the Teradata server that says that the reference is ambiguous. If the temporary table name were named FOO\_TMP1, then the collision error would not have happened.

# Passing SAS Functions to Teradata

SAS/ACCESS Interface to Teradata passes the following SAS functions to Teradata for processing. Where the Teradata function name differs from the SAS function name, the Teradata name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                  |                   |
|------------------|-------------------|
| ABS              | MAX               |
| ACOS             | MIN               |
| ARCOSH (ACOSH)   | MINUTE            |
| ARSINH (ASINH)   | MOD (see note)    |
| ASIN             | MONTH             |
| ATAN             | SECOND            |
| ATAN2            | SIN               |
| Avg              | SINH              |
| COALESCE         | SQRT              |
| COS              | STD (STDDEV_SAMP) |
| COSH             | STRIP (TRIM)      |
| COUNT            | SUBSTR            |
| DATEPART         | TAN               |
| DAY              | TANH              |
| EXP              | TIMEPART          |
| HOUR             | TRIM              |
| INDEX (POSITION) | UPCASE            |
| LOG              | VAR (VAR_SAMP)    |
| LOG10            | YEAR              |
| LOWCASE (LCASE)  |                   |

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can SAS/ACCESS Interface to Teradata also pass these SAS SQL functions to Teradata. Due to incompatibility in date and time functions between Teradata and SAS, Teradata might not process them correctly. Check your results to determine whether these functions are working as expected. For more information, see [SQL\\_FUNCTIONS=LIBNAME Option on page 324](#).

|                              |                     |
|------------------------------|---------------------|
| DATE                         | SOUNDEX             |
| DATETIME (current_timestamp) | TIME (current_time) |
| LEFT (TRIM)                  | TODAY               |

#### LENGTHC (CHARACTER\_LENGTH)

DATETIME, SOUNDEX, and TIME are not entirely compatible with the corresponding SAS functions. Also, for SOUNDEX, although Teradata always returns 4 characters, SAS might return more or less than 4 characters.

---

## Passing Joins to Teradata

For a multiple libref join to pass to Teradata, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- account ID (ACCOUNT=)
- server (TDPID= or SERVER=)

You must specify the SCHEMA= LIBNAME option to fully qualify each table name in a join for each LIBNAME that you reference.

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Maximizing Teradata Load and Read Performance

---

### Overview

SAS/ACCESS supports the [Teradata Parallel Transporter application programming interface](#) (TPT API). TPT is now the DEFAULT action for UNIX and PC hosts. FastLoad and MultiLoad are considered to be legacy features on UNIX and PC hosts. For more information, see the Teradata Parallel Transporter documentation.

---

**Note:** If for some reason the TPT method is not available, SAS/ACCESS automatically uses FastLoad, FastExport, or MultiLoad methods to insert or retrieve data. By default, you are notified in the SAS log that an alternative method was used. If you want to suppress these messages, set the SAS\_TPT\_SHOW\_MSGS environment variable to NO before you start SAS.

---

---

## Using the TPT API

---

### TPT API Supported Features and Restrictions

SAS/ACCESS Interface to Teradata supports the TPT API for loading and reading data. The TPT API provides a consistent interface for Fastload, FastExport, MultiLoad, and Multi-Statement insert. TPT API documentation refers to Fastload as the *Load operator*, FastExport as the *Export operator*, MultiLoad as the *Update operator*, and Multi-Statement insert as the *Stream operator*. Restart functionality is supported by Fastload, MultiLoad, and Multi-Statement insert.

---

### TPT API Setup

Here are the requirements for using the TPT API in SAS for loading SAS.

- Loading data from SAS to Teradata using the TPT API requires Teradata client TTU 8.2 or later. Verify that you have applied all of the latest Teradata eFixes.
- This feature is supported only on platforms for which Teradata provides the TPT API.
- The native TPT API infrastructure must be present on your system. Contact Teradata if you do not already have it but want to use it with SAS.
- These Teradata privileges are needed to load a Teradata table using the TPT API. For more information, see the Teradata TPT API documentation.
  - CREATE TABLE
  - DROP TABLE
  - CREATE MACRO (Multi-Statement Stream operator)
  - DROP MACRO (Multi-Statement Stream operator)
  - INSERT
  - DELETE
  - SELECT

The SAS configuration document for your system contains information about how to configure SAS to work with the TPT API. However, those steps might already have been completed as part of the post-installation configuration process for your site.

---

### TPT API LIBNAME Options

These LIBNAME options are common to all three supported load methods:

- LOGDB=
- SLEEP=
- TENACITY=
- TPT=
- TPT\_DATA\_ENCRYPTION=
- TPT\_MAX\_SESSIONS=
- TPT\_MIN\_SESSIONS=
- TPT\_UNICODE\_PASSTHRU=

If SAS cannot use the TPT API, it reverts to using Fastload, MultiLoad, or Multi-Statement insert, depending on which method of loading was requested without generating any errors.

## TPT API Data Set Options

These data set options are common to all three supported load methods:

- SLEEP=
- TENACITY=
- TPT=
- TPT\_APPL\_PHASE=
- TPT\_BLOCK\_SIZE=
- TPT\_BUFFER\_SIZE=
- TPT\_CHECKPOINT\_DATA=
- TPT\_DATA\_ENCRYPTION=
- TPT\_ERROR\_TABLE\_1=
- TPT\_ERROR\_TABLE\_2=
- TPT\_LOG\_TABLE=
- TPT\_MAX\_SESSIONS=
- TPT\_MIN\_SESSIONS=
- TPT\_PACK=
- TPT\_PACKMAXIMUM=
- TPT\_RESTART=
- TPT\_TRACE\_LEVEL=
- TPT\_TRACE\_LEVEL\_INF=
- TPT\_TRACE\_OUTPUT=
- TPT\_WORK\_TABLE=

You can use the DBCommit= LIBNAME option and the CHECKPOINT= data set option to specify checkpoint frequency with TPT for Fastload, MultiLoad, and Multi-Statement insert. These options are disabled only for the non-TPT MultiLoad

interface and are not relevant with TPT FastExport. Note that SAS sets TPT\_MAX\_SESSIONS to 4.

---

## Processing Large Response Rows

SAS/ACCESS automatically verifies that your Teradata environment is configured to process response row sizes up to 1MB. Typically, the required versions of the Teradata client libraries, Teradata TTU, and Teradata Vantage SQL Engine all support row sizes of 1MB.

The Teradata legacy utilities—FastLoad, MultiLoad, MultiStatement, and FastExport—all support row sizes up to 1MB as long as the LIBNAME option TPT= is set to YES (the default). If TPT=NO, then the legacy utilities process rows only up to a maximum size of 64K.

If you use the default values for TPT related options, then the maximum row size is 1MB. You can restrict the maximum row size to 64K by setting TPT=NO or by setting the TD\_1MB\_ROW environment variable to OFF. The default value for TD\_1MB\_ROW is ON. For more information, see “[TD\\_1MB\\_ROW Environment Variable](#)” in *SAS Global Statements: Reference*.

---

## TPT API FastLoad Supported Features and Restrictions

SAS/ACCESS Interface to Teradata supports the TPT API for FastLoad, also known as the *Load operator*. SAS/ACCESS works by interfacing with the Load operator through the TPT API, which in turn uses the Teradata Fastload protocol for loading data. See your Teradata documentation for more information about the Load operator.

This is the default FastLoad method. If SAS cannot find the Teradata modules that are required for the TPT API or if TPT=NO, SAS/ACCESS uses the old method of Fastload. SAS/ACCESS can restart Fastload from checkpoints when FastLoad uses the TPT API. The SAS/ACCESS FastLoad facility using the TPT API is similar to the native Teradata FastLoad utility. They share these limitations.

- FastLoad can load only empty tables. It cannot append to a table that already contains data. If you try to use FastLoad when appending to a table that contains rows, the append step fails.
- Data errors are logged in Teradata tables. Error recovery can be difficult if you do not set TPT\_CHECKPOINT\_DATA= to enable restart from the last checkpoint. To find the error that corresponds to the code that is stored in the error table, see your Teradata documentation. You can restart a failed job for the last checkpoint by following the instructions in the SAS error log.
- FastLoad does not load duplicate rows (those where all corresponding fields contain identical data) into a Teradata table. If your SAS data set contains duplicate rows, you can use other load methods.

---

## Starting FastLoad with the TPT API

See the SAS configuration document for instructions on setting up the environment so that SAS can find the TPT API modules.

You can use one of these options to start FastLoad in the SAS/ACCESS interface using the TPT API:

- the TPT=YES data set option in a processing step that populates an empty Teradata table
- the TPT=YES LIBNAME option on the destination libref (the Teradata DBMS library where one or more tables are to be created and loaded)

---

## FastLoad with TPT API Data Set Options

These data set options are specific to FastLoad using the TPT API:

- [TPT\\_BUFFER\\_SIZE=](#)
- [TPT\\_DATA\\_ENCRYPTION=](#)
- [TPT\\_ERROR\\_TABLE\\_1=](#)
- [TPT\\_ERROR\\_TABLE\\_2=](#)

---

## TPT API MultiLoad Supported Features and Restrictions

SAS/ACCESS Interface to Teradata supports the TPT API for MultiLoad, also known as the *Update operator*. SAS/ACCESS works by interfacing with the Update operator through the TPT API. This API then uses the Teradata Multiload protocol for loading data. See your Teradata documentation for more information about the Update operator.

This is the default MultiLoad method. If SAS cannot find the Teradata modules that are required for the TPT API or TPT=NO, SAS/ACCESS uses the old method of MultiLoad. SAS/ACCESS can restart MultiLoad from checkpoints when MultiLoad uses the TPT API.

The SAS/ACCESS MultiLoad facility loads both empty and existing Teradata tables. SAS/ACCESS supports only Insert operations and loading only one target table at a time.

The SAS/ACCESS MultLoad facility using the TPT API is similar to the native Teradata MultiLoad utility. A common limitation that they share is that you must drop these items onto target tables before the load:

- unique secondary indexes
- foreign key references

- join indexes

Errors are logged to Teradata tables. Error recovery can be difficult if you do not set **TPT\_CHECKPOINT\_DATA=** to enable restart from the last checkpoint. To find the error that corresponds to the code that is stored in the error table, see your Teradata documentation. You can restart a failed job for the last checkpoint by following the instructions in the SAS error log.

## Starting MultiLoad with the TPT API

See the SAS configuration document for instructions on setting up the environment so that SAS can find the TPT API modules.

You can use one of these options to start MultiLoad in the SAS/ACCESS interface using the TPT API:

- the TPT=YES data set option in a processing step that populates an empty Teradata table
- the TPT=YES LIBNAME option on the destination libref (the Teradata DBMS library where one or more tables are to be created and loaded)

## MultiLoad with TPT API LIBNAME Options

These LIBNAME options are specific to MultiLoad using the TPT API:

- **LOGDB=**
- **TPT\_DATA\_ENCRYPTION=**
- **TPT\_UNICODE\_PASSTHRU=**

## MultiLoad with TPT API Data Set Options

These data set options are specific to MultiLoad using the TPT API:

- **TPT\_BUFFER\_SIZE=**
- **TPT\_DATA\_ENCRYPTION=**
- **TPT\_ERROR\_TABLE\_1=**
- **TPT\_ERROR\_TABLE\_2=**

---

## TPT API Multi-Statement Insert Supported Features and Restrictions

SAS/ACCESS Interface to Teradata supports the TPT API for Multi-Statement insert, also known as the *Stream operator*. SAS/ACCESS works by interfacing with the Stream operator through the TPT API, which in turn uses the Teradata Multi-Statement insert (TPump) protocol for loading data. See your Teradata documentation for more information about the Stream operator.

This is the default Multi-Statement insert method. If SAS cannot find the Teradata modules that are required for the TPT API or TPT=NO, SAS/ACCESS uses the old method of Multi-Statement insert. SAS/ACCESS can restart Multi-Statement insert from checkpoints when Multi-Statement insert uses the TPT API.

The SAS/ACCESS Multi-Statement insert facility loads both empty and existing Teradata tables. SAS/ACCESS supports only Insert operations and loading only one target table at time.

Errors are logged to Teradata tables. Error recovery can be difficult if you do not set TPT\_CHECKPOINT\_DATA= to enable restart from the last checkpoint. To find the error that corresponds to the code that is stored in the error table, see your Teradata documentation. You can restart a failed job for the last checkpoint by following the instructions on the SAS error log.

---

## Starting Multi-Statement Insert with the TPT API

See the SAS configuration document for instructions on setting up the environment so that SAS can find the TPT API modules.

You can use one of these options to start Multi-Statement in the SAS/ACCESS interface using the TPT API:

- the TPT=YES data set option in a processing step that populates an empty Teradata table
- the TPT=YES LIBNAME option on the destination libref (the Teradata DBMS library where one or more tables are to be created and loaded)

These LIBNAME options also work with the Multi-Statement Insert:

- [TPT\\_DATA\\_ENCRYPTION=](#)
- [TPT\\_UNICODE\\_PASSTHRU=](#)

## Multi-Statement Insert with TPT API Data Set Options

These data set options are specific to Multi-Statement insert using the TPT API.

- [TPT\\_DATA\\_ENCRYPTION=](#)
- [TPT\\_PACK=](#)
- [TPT\\_PACKMAXIMUM=](#)

## Legacy Load and Read Utilities

If the TPT API is not available, SAS/ACCESS Interface to Teradata provides the following facilities. Although these are legacy facilities, they do improve performance when loading data. These correspond to native Teradata utilities.

- [FastLoad](#)
- [FastExport](#)
- [MultiLoad](#)
- [Multi-Statement](#)

For more information, see the following sections.

## Using FastLoad

### FastLoad Supported Features and Restrictions

SAS/ACCESS Interface to Teradata supports FastLoad, a native Teradata bulk-load utility that greatly accelerates inserting data into empty Teradata tables. For general information about using FastLoad and error recovery, see the Teradata FastLoad documentation.

The SAS/ACCESS FastLoad facility is similar to the native Teradata FastLoad Utility. They share these limitations:

- FastLoad can load only empty tables. It cannot append to a table that already contains data. If you try to use FastLoad when appending to a table that contains rows, the append step fails.
- Both the Teradata FastLoad Utility and the SAS/ACCESS FastLoad facility log data errors to tables. Error recovery can be difficult. To find the error that

corresponds to the code that is stored in the error table, see the Teradata FastLoad documentation.

- FastLoad does not load duplicate rows (rows where all corresponding fields contain identical data) into a Teradata table. If your SAS data set contains duplicate rows, you can use the normal insert (load) process.
- If you create a table with special characters that require quotation marks, you must specify a name in BL\_LOG= that does not require quotation marks.

---

## Starting FastLoad

If you do not specify FastLoad, your Teradata tables are loaded normally (slowly). To start FastLoad in the SAS/ACCESS interface, you can use one of these items:

- the BULKLOAD=YES data set option in a processing step that populates an empty Teradata table
- the BULKLOAD=YES LIBNAME option on the destination libref (the Teradata DBMS library where one or more intended tables are to be created and loaded)
- the FASTLOAD= alias for either of these options

---

## FastLoad Data Set Options

Here are the data set options that you can use with the FastLoad facility.

- **BL\_LOG=** specifies the names of error tables that are created when you use the SAS/ACCESS FastLoad facility. By default, FastLoad errors are logged in Teradata tables named SAS\_FASTLOAD\_ERRS1\_<random> and SAS\_FASTLOAD\_ERRS2\_<random>, where <random> is a randomly generated number. For example, if you specify `BL_LOG=my_load_errors`, errors are logged in tables `my_load_errors1` and `my_load_errors2`.

**Note:** SAS/ACCESS automatically deletes the error tables if no errors are logged. If errors occur, the tables are retained and SAS/ACCESS issues a warning message that includes the names of the error tables.

- **DBCOMMIT=n** causes a Teradata “checkpoint” after each group of *n* rows is transmitted. Using checkpoints slows performance but provides known synchronization points if failure occurs during the loading process. Checkpoints are not used by default if you do not explicitly specify DBCOMMIT= and BULKLOAD=YES. The Teradata alias for this option is CHECKPOINT=.

For details, see “[About the Data Set Options for Relational Databases](#)” on page 370.

---

## Using MultiLoad

---

### MultiLoad Supported Features and Restrictions

SAS/ACCESS Interface to Teradata supports a bulk-load capability called MultiLoad that greatly accelerates insertion of data into Teradata tables. For general information about using MultiLoad with Teradata tables and for information about error recovery, see the Teradata MultiLoad documentation. SAS/ACCESS examples are available.

Unlike FastLoad, which loads only empty tables, MultiLoad loads both empty and existing Teradata tables. If you do not specify MultiLoad, your Teradata tables are loaded normally (inserts are sent one row at a time).

The SAS/ACCESS MultiLoad facility loads both empty and existing Teradata tables. SAS/ACCESS supports these features:

- You can load only one target table at a time.
- Only Insert and Upsert operations are supported.

Because the SAS/ACCESS MultiLoad facility is similar to the native Teradata MultiLoad utility, they share a limitation: You must drop these items onto the target tables before the load:

- unique secondary indexes
- foreign key references
- join indexes

Both the Teradata MultiLoad utility and the SAS/ACCESS MultiLoad facility log data errors to tables. Error recovery can be difficult, but the ability to restart from the last checkpoint is possible. To find the error that corresponds to the code that is stored in the error table, see the Teradata MultiLoad documentation.

---

### MultiLoad Setup

Here are the requirements for using the MultiLoad bulk-load capability in SAS.

- The native Teradata MultiLoad utility must be present on your system. If you do not have the Teradata MultiLoad utility and you want to use it with SAS, contact Teradata to obtain the utility.
- SAS must be able to locate the Teradata MultiLoad utility on your system.
- The Teradata MultiLoad utility must be able to locate the SASMlasm access module and the SasMlne exit routine. They are supplied with SAS/ACCESS Interface to Teradata.
- SAS MultiLoad requires Teradata client TTU 8.2 or later.

If not already done during post-installation configuration, see the SAS configuration documentation for your system for information about how to configure SAS to work with MultiLoad.

---

## MultiLoad Data Set Options

Call the SAS/ACCESS MultiLoad facility by specifying MULTILOAD=YES. See the MULTILOAD= data set option for detailed information and examples on loading data and recovering from errors during the load process.

Here are the data set options that are available for use with the MultiLoad facility. For detailed information about these options, see [“Overview” on page 370](#).

- **MBUFSIZE=**
- **ML\_CHECKPOINT=**
- **ML\_ERROR1=** allows the user name the error table that MultiLoad uses for tracking errors from the acquisition phase. See the Teradata MultiLoad reference for more information about what is stored in this table. By default, the acquisition error table is named SAS\_ML\_ET\_<random> where <random> is a random number. When restarting a failed MultiLoad job, you need to specify the same acquisition table from the earlier run so that the MultiLoad job can restart correctly. Note that the same log table, application error table, and work table must also be specified upon restarting, using ML\_RESTART, ML\_ERROR2, and ML\_WORK data set options. ML\_ERROR1 and ML\_LOG are mutually exclusive and cannot be specified together.
- **ML\_ERROR2=**
- **ML\_LOG=** specifies a prefix for the temporary tables that the Teradata MultiLoad utility uses during the load process. The MultiLoad utility uses a log table, two error tables, and a work table when loading data to the target table. These tables are named by default as SAS\_ML\_RS\_<random>, SAS\_ML\_ET\_<random>, SAS\_ML\_UT\_<random>, and SAS\_ML\_WT\_<random> where <random> is a randomly generated number. ML\_LOG= is used to override the default names used. For example, if you specify ML\_LOG=MY\_LOAD the log table is named MY\_LOAD\_RS. Errors are logged in tables MY\_LOAD\_ET and MY\_LOAD\_UT. The work table is named MY\_LOAD\_WT.
- **ML\_RESTART=** allows the user name the log table that MultiLoad uses for tracking checkpoint information. By default, the log table is named SAS\_ML\_RS\_<random> where <random> is a random number. When restarting a failed MultiLoad job, you need to specify the same log table from the earlier run so that the MultiLoad job can restart correctly. Note that the same error tables and work table must also be specified upon restarting the job, using ML\_ERROR1, ML\_ERROR2, and ML\_WORK data set options. ML\_RESTART and ML\_LOG are mutually exclusive and cannot be specified together.
- **ML\_WORK=** allows the user name the work table that MultiLoad uses for loading the target table. See the Teradata MultiLoad reference for more information about what is stored in this table. By default, the work table is named SAS\_ML\_WT\_<random> where <random> is a random number. When restarting a failed MultiLoad job, you need to specify the same work table from the earlier run so that the MultiLoad job can restart correctly. Note that the same log table, acquisition error table and application error table must also be specified upon restarting the job using ML\_RESTART, ML\_ERROR1, and ML\_ERROR2 data

set options. ML\_WORK and ML\_LOG are mutually exclusive and cannot be specified together.

- **SLEEP=** specifies the number of minutes that MultiLoad waits before it retries a logon operation when the maximum number of utilities are already running on the Teradata database. The default value is 6. SLEEP= functions very much like the SLEEP run-time option of the native Teradata MultiLoad utility.
- **TENACITY=** specifies the number of hours that MultiLoad tries to log on when the maximum number of utilities are already running on the Teradata database. The default value is 4. TENACITY= functions very much like the TENACITY run-time option of the native Teradata MultiLoad utility.
- **UPSERT=** when MultiLoad=YES, specifies that a Teradata Upsert operation should take place. The default value is NO.
- **UPSERT\_WHERE=** specifies which columns in the master table are to be used when generating a Where condition for a MultiLoad Upsert. The default value is none.
- **UPSERT\_CONDITION=** specifies additional conditions to be appended to the UPSERT\_WHERE= option. The default value is none.

Be aware that these options are disabled while you are using the SAS/ACCESS MultiLoad facility.

- The **DBCOMMIT= LIBNAME** and data set options are disabled because DBCOMMIT= functions very differently from CHECKPOINT of the native Teradata MultiLoad utility.
- A rollback does not take place to the last checkpoint on reaching **ERRLIMIT=** as the rows without errors have already been sent to Teradata.

To see whether threaded Reads are actually generated, turn on SAS tracing by specifying **OPTIONS SASTRACE=" , , , d"** in your program.

## MultiLoad Examples

This example starts the FastLoad facility.

```
libname fload teradata user=myusr1 password=mypwd1;
data fload.nffloat(bulkload=yes);
do x=1 to 1000000;
output;
end;
run;
```

This next example uses FastLoad to append SAS data to an empty Teradata table and specifies the **BL\_LOG=** option to name the error tables **Append\_Err1** and **Append\_Err2**. In practice, applications typically append many rows.

```
/* Create the empty Teradata table. */
proc sql;
connect to teradata as tera(user=myusr1 password=mypwd1);
execute (create table performers
(userid int, salary decimal(10,2), job_desc char(50)))
by tera;
execute (commit) by tera;
quit;
```

```

/* Create the SAS data to load. */
data local;
    input userid 5. salary 9. job_desc $50. ;
    datalines;
        0433 35993.00 grounds keeper
        4432 44339.92 code groomer
        3288 59000.00 manager
    ;

/* Append the SAS data & name the Teradata error tables. */
libname tera teradata user=myusr1 password=mypwd1;

proc append data=local base=tera.performers
    (bulkload=yes bl_log=append_err);
run;

```

This example starts the MultiLoad facility.

```

libname trlib teradata user=myusr1 pw=mypwd1 server=dbc;

/* Use MultiLoad to load a table with 2000 rows. */
data trlib.mlfload(MultiLoad=yes);
    do x=1 to 2000;
        output;
    end;
run;

/* Append another 1000 rows. */
data worktestdata;
    do x=2001 to 3000;
        output;
    end;
run;

/* Append the SAS data to the Teradata table. */
proc append data=worktestdata base=trlib.mlfload
    (MultiLoad=yes);
run;

```

This example loads data using TPT FastLoad.

```

/* Check the SAS log for this message to verify that the TPT API was
used.
NOTE: Teradata connection: TPT Fastload has inserted 100 rows.
*/
data trlib.load(TPT=YES FASTLOAD=YES);
    do x=1 to 1000;
        output;
    end;
run;

```

This example restarts a MultiLoad step that recorded checkpoints and failed after loading 2000 rows of data.

```

proc append data=trlib.load(TPT=YES MULTILOAD=YES
    TPT_RESTART=YES TPT_CHECKPOINT_DATA=2000)
    data=work.inputdata(FIRSTOBS=2001);
run;

```

---

# Autopartitioning Scheme for Teradata (Legacy)

---

## Overview

For general information about this feature, see “[Autopartitioning Techniques in SAS/ACCESS](#)” on page [76](#). For platform-specific details and special considerations, see your Teradata documentation.

The FastExport Utility is the fastest available way to read large Teradata tables. FastExport is software that Teradata provides that delivers data over multiple Teradata connections or sessions. If FastExport is available, SAS threaded Reads use it. If FastExport is not available, SAS threaded Reads generate partitioning WHERE clauses. Using the DBSLICE= option overrides FastExport. So if you have FastExport available and want to use it, do not use DBSLICE=. To use FastExport everywhere possible, use DBSLICEPARM=ALL.

---

**Note:** Whether automatically generated or created by using DBSLICE=, partitioning of WHERE clauses is not supported.

---

---

## FastExport and Case Sensitivity

In certain situations Teradata returns different row results to SAS when using FastExport, compared to reading normally without FastExport. The difference arises only when all of these conditions are met:

- A WHERE clause that compares a character column with a character literal is asserted.
- The column definition is NOT CASESPECIFIC.

Unless you specify otherwise, most Teradata native utilities create NOT CASESPECIFIC character columns. On the other hand, SAS/ACCESS Interface to Teradata creates CASESPECIFIC columns. In general, this means that you do not see result differences with tables that SAS creates. However, you might see result differences with tables that Teradata utilities create, which are frequently many of your tables. To determine how Teradata creates a table, look at your column declarations with the Teradata SHOW TABLE statement.

- A character literal matches to a column value that differs only in case.

You can see differences in the rows returned if your character column has mixed-case data that is otherwise identical. For example, 'Top' and 'top' are identical except for case.

Case sensitivity is an issue when SAS generates SQL code that contains a WHERE clause with one or more character comparisons. It is also an issue when you provide the Teradata SQL yourself with the explicit SQL feature of PROC SQL. The following examples illustrate each scenario, using DBSICEPARM=ALL to start FastExport instead of the normal SAS read:

```
/* SAS generates the SQL for you. */
libname trlible teradata user=myusr1 password=mypwd1 dbsliceparm=all;
proc print data=trlible.employees;
where lastname='lovell';
run;

/* Use explicit SQL with PROC SQL & provide the
SQL yourself, also starting FastExport. */
proc sql;
connect to teradata(user=myusr1 password=mypwd1 dbsliceparm=all);
select * from connection to teradata
(select * from sales where gender='f' and salesamt>1000);
quit;
```

For more information about case sensitivity, see your Teradata documentation.

## FastExport Password Security

FastExport requires passwords to be in clear text. Because this poses a security risk, users must specify the full path name so that the file path is in a protected directory:

- Windows users should specify *BL\_CONTROL="PROTECTED-DIR/myscr.ctl"*. SAS/ACCESS creates the myscr.ctl script file in the protected directory with PROTECTED-DIR as the path.
- UNIX users can specify a similar path name.
- Users with z/OS must specify a middle-level qualifier such as *BL\_CONTROL="MYSCR.TEST1"* so that the system generates the *USERID.MYSCR.TEST1.CTL* script file.
- Users can also use RACF to protect the *USERID.MYSCR\** profile.

## FastExport Setup

There are three requirements for using FastExport with SAS:

- You must have the Teradata FastExport utility present on your system. If you do not have FastExport and want to use it with SAS, contact Teradata to obtain this utility.
- SAS must be able to locate the FastExport Utility on your system.
- The FastExport Utility must be able to locate the SasAxsm access module, which is supplied with your SAS/ACCESS Interface to Teradata product. SasAxsm is in the SAS directory tree, in the same location as the Teradata component.

Assuming you have the Teradata FastExport Utility, perform this setup, which varies by system:

- **Windows:** As needed, modify your Path environment variable to include *both* the directories containing Fexp.exe (FastExport) and SasAxsm. Place these directory specifications last in your path.
- **UNIX:** As needed, modify your library path environment variable to include the directory containing sasaxsm.sl (HP) or sasaxsm.so (Solaris and AIX). These shared objects are delivered in the \$SASROOT/sasexe directory. You can copy these modules where you want, but make sure that the directory into which you copy them is in the appropriate shared library path environment variable. On Solaris, the library path variable is LD\_LIBRARY\_PATH. On HP-UX, it is SHLIB\_PATH. On AIX, it is LIBPATH. Also, make sure that the directory containing the Teradata FastExport Utility (fexp), is included in the PATH environment variable. FastExport is usually installed in the /usr/bin directory.
- **z/OS:** No action is needed when starting FastExport under TSO. When starting FastExport with a batch JCL, the SAS source statements must be assigned to a DD name other than SYSIN. This can be done by passing a parameter such as SYSIN=SASIN in the JCL where all SAS source statements are assigned to the DD name SASIN.

Keep in mind that future releases of SAS might require an updated version of SasAxsm. Therefore, when upgrading to a new SAS version, you should update the path for SAS on Windows and the library path for SAS on UNIX.

## Using FastExport

To use FastExport, SAS writes a specialized script to a disk that the FastExport Utility reads. SAS might also log FastExport log lines to another disk file. SAS creates and deletes these files on your behalf, so no intervention is required. Sockets deliver the data from FastExport to SAS, so you do not need to do anything except install the SasAxsm access module that enables data transfer.

On Windows, when the FastExport Utility is active, a DOS window appears minimized as an icon on your toolbar. You can maximize the DOS window, but do not close it. After a FastExport operation is complete, SAS closes the window for you.

This example shows how to create a SAS data set that is a subset of a Teradata table that uses FastExport to transfer the data:

```
libname trlib teradata user=myusr1 password=mypwd1;
data sasllocal(keep=EMPID SALARY);
  set trlib.employees(dbsliceparm=all);
run;
```

## FastExport and Explicit SQL

FastExport is also supported for the explicit SQL feature of PROC SQL.

The following example shows how to create a SAS data set that is a subset of a Teradata table by using explicit SQL and FastExport to transfer the data:

```

proc sql;
connect to teradata as pro1 (user=myusr1 password=mypwd1
dbsliceparm=all);
create table saslocal as select * from connection to pro1
(select EMPID, SALARY from employees);
quit;

```

FastExport for explicit SQL is a Teradata extension only, for optimizing Read operations, and is not covered in the threaded Read documentation.

## Exceptions to Using FastExport

With the Teradata FastExport Utility and the SasAxsm module in place that SAS supplies, FastExport works automatically for all SAS steps where threaded Reads are enabled, except for one situation. FastExport does not handle single Access Module Processor (AMP) queries. In this case, SAS/ACCESS simply reverts to a normal single connection read. For information about FastExport and single AMP queries, see your Teradata documentation.

To determine whether FastExport worked, turn on SAS tracing in advance of the step that attempts to use FastExport. If you use FastExport, you receive this (English only) message, which is written to your SAS log:

```
teradata/tryottrm(): SELECT was processed with FastExport.
```

To turn on SAS tracing, run this statement:

```
options sastrace=',,d' sastraceloc=saslog;
```

## Threaded Reads with Partitioning WHERE Clauses

If FastExport is unavailable, threaded Reads use partitioning WHERE clauses. You can create your own partitioning WHERE clauses using the DBSLICE= option. Otherwise, SAS/ACCESS to Teradata attempts to generate them on your behalf. Like other SAS/ACCESS interfaces, this partitioning is based on the MOD function. To generate partitioning WHERE clauses, SAS/ACCESS to Teradata must locate a table column suitable for applying MOD. These types are eligible:

- BYTEINT
- SMALLINT
- INTEGER
- DATE
- DECIMAL (integral DECIMAL columns only)

A DECIMAL column is eligible only if the column definition restricts it to integer values. In other words, the DECIMAL column must be specified with a scale of zero.

If the table that you are reading contains more than one column of the above mentioned types, SAS/ACCESS to Teradata applies some nominal intelligence to select a best choice. Top priority is given to the primary index, if it is MOD-eligible. Otherwise, preference is given to any column that is specified as NOT NULL.

Because this is an unsophisticated set of selection rules, you might want to provide your own partitioning using the DBSLICE= option.

To view your table's column definitions, use the Teradata SHOW TABLE statement.

---

**Note:** Partitioning WHERE clauses, either automatically generated or created by using DBSLICE=, are not supported on z/OS. Whether automatically generated or created by using DBSLICE=, partitioning WHERE clauses is not supported on z/OS and UNIX.

---

## FastExport versus Partitioning WHERE Clauses

Partitioning WHERE clauses are innately less efficient than FastExport. The Teradata DBMS must process separate SQL statements that vary in the WHERE clause. In contrast, FastExport is optimal because only one SQL statement is transmitted to the Teradata DBMS. However, older editions of the Teradata DBMS place severe restrictions on the system-wide number of simultaneous FastExport operations that are allowed. Even with newer versions of Teradata, your database administrator might be concerned about large numbers of FastExport operations.

Threaded Reads with partitioning WHERE clauses also place higher workload on Teradata and might not be appropriate on a widespread basis. Both technologies expedite throughput between SAS and the Teradata DBMS but should be used judiciously. For this reason, only SAS threaded applications are eligible for threaded Read by default. To enable more threaded Reads or to turn them off entirely, use the DBSLICEPARM= option.

Even when FastExport is available, you can force SAS/ACCESS to Teradata to generate partitioning WHERE clauses on your behalf. This is accomplished with the DBI argument to the DBSLICEPARM= option (DBSLICEPARM=DBI). This feature is available primarily to enable comparisons of these techniques. In general, you should use FastExport if it is available.

The explicit SQL feature of PROC SQL supports FastExport. Partitioning WHERE clauses is not supported for explicit SQL.

---

## Teradata Processing Tips for SAS Users

---

### Reading from and Inserting to the Same Teradata Table

If you use SAS/ACCESS to read rows from a Teradata table and then attempt to insert these rows into the same table, you can hang (suspend) your SAS session.

Here is what happens:

- A SAS/ACCESS connection requests a standard Teradata READ lock for the Read operation.
- A SAS/ACCESS connection then requests a standard Teradata WRITE lock for the Insert operation.
- The WRITE lock request suspends because the read connection already holds a READ lock on the table. Consequently, your SAS session stops responding (is suspended).

Here is what happens in the next example:

- SAS/ACCESS creates a read connection to Teradata to fetch the rows selected (select \*) from TRA.SAMETABLE, requiring a standard Teradata READ lock; Teradata issues a READ lock.
- SAS/ACCESS creates an insert connection to Teradata to insert the rows into TRA.SAMETABLE, requiring a standard Teradata WRITE lock. But the WRITE lock request suspends because the table is locked already by the READ lock.
- Your SAS/ACCESS session stops responding.

```
libname tra teradata user=myusr1 password=mypwd1;
proc sql;
insert into tra.sametable
    select * from tra.sametable;
```

To avoid this situation, use the SAS/ACCESS locking options. For details, see [“Locking in the Teradata Interface” on page 1363](#).

## Using a BY Clause to Order Query Results

SAS/ACCESS returns table results from a query in random order because Teradata returns the rows to SAS/ACCESS randomly. In contrast, traditional SAS processing returns SAS data set observations in the same order during every run of your job. If maintaining row order is important, you should add a BY clause to your SAS statements. A BY clause ensures consistent ordering of the table results from Teradata.

In this example, the Teradata ORD table has NAME and NUMBER columns. The PROC PRINT statements illustrate consistent and inconsistent ordering when it displays ORD table rows.

```
libname prt teradata user=myusr1 password=mypwd1;

proc print data=prt.ORD;
  var name number;
run;
```

If this statement is run several times, it yields inconsistent ordering, meaning that ORD rows are likely to be arranged differently each time. This happens because SAS/ACCESS displays the rows in the order in which Teradata returns them—that is, randomly.

```
proc print data=prt.ORD;
  var name number;
  by name;
run;
```

This statement achieves more consistent ordering because it orders PROC PRINT output by the NAME value. However, on successive runs of the statement, rows display of rows with a different number and an identical name can vary, as shown here.

**Output 39.1 PROC PRINT Display 1**

```
Rita Calvin 2222
Rita Calvin 199
```

**Output 39.2 PROC PRINT Display 2**

```
Rita Calvin 199
Rita Calvin 2222
```

```
proc print data=prt.ORD;
var name number;
by name number;
run;
```

The above statement always yields identical ordering because every column is specified in the BY clause. Your PROC PRINT output always looks the same.

## Using TIME and TIMESTAMP

This example creates a Teradata table and assigns the SAS TIME8. format to the TRXTIME0 column. Teradata creates the TRXTIME0 column as the equivalent Teradata data type, TIME(0), with the value of 12:30:55.

```
libname mylib teradata user=myusr1 password=mypwd1;

data mylib.trxtimes;
format trxtime0 time8.;
trxtime0 = '12:30:55't;
run;
```

This example creates a Teradata column that specifies very precise time values. The format TIME(5) is specified for the TRXTIME5 column. When SAS reads this column, it assigns the equivalent SAS format TIME14.5.

```
libname mylib teradata user=myusr1 password=mypwd1;

proc sql noerrorstop;
connect to teradata (user=myusr1 password=mypwd1);
execute (create table trxtimes (trxtime5 time(5)
)) by teradata;
execute (commit) by teradata;
execute (insert into trxtimes
values (cast('12:12:12' as time(5)))
) by teradata;
execute (commit) by teradata;
quit;
```

```
/* You can print the value that is read with SAS/ACCESS. */
proc print data =mylib.trxtimes;
run;
```

SAS might not preserve more than four digits of fractional precision for Teradata TIMESTAMP.

This next example creates a Teradata table and specifies a simple timestamp column with no digits of precision. Teradata stores the value 2000-01-01 00:00:00. SAS assigns the default format DATETIME19. to the TRSTAMP0 column generating the corresponding SAS value of 01JAN2000:00:00:00.

```
proc sql noerrorstop;
connect to teradata (user=myusr1 password=mypwd1);
execute (create table stamps (tstamp0 timestamp(0)
)) by teradata;
execute (commit) by teradata;
execute (insert into stamps
values (cast('2000-01-01 00:00:00' as
timestamp(0)))
)) by teradata;
execute (commit) by teradata;
quit;
```

This example creates a Teradata table and assigns the SAS format DATETIME23.3 to the TSTAMP3 column, generating the value 13APR1961:12:30:55.123. Teradata creates the TSTAMP3 column as the equivalent data type TIMESTAMP(3) with the value 1961-04-13 12:30:55.123.

```
libname mylib teradata user=myusr1 password=mypwd1;

data mylib.stamps;
format tstamp3 datetime23.3;
tstamp3 = '13apr1961:12:30:55.123'dt;
run;
```

This next example illustrates how the SAS engine passes the literal value for TIMESTAMP in a WHERE statement to Teradata for processing. Note that the value is passed without being rounded or truncated so that Teradata can handle the rounding or truncation during processing. This example would also work in a DATA step.

```
proc sql ;
select * from trlib.flytime where col1 = '22Aug1995 12:30:00.557'dt ;
quit;
```

## Replacing PROC SORT with a BY Clause

In general, PROC SORT steps are not useful to export a Teradata table. In traditional SAS processing, PROC SORT is used to order observations in a SAS data set. Subsequent SAS steps that use the sorted data set receive and process the observations in the sorted order. Teradata does not store output rows in the sorted order. Therefore, do not sort rows with PROC SORT if the destination sorted file is a Teradata table.

This example illustrates a PROC SORT statement found in typical SAS processing. You cannot use this statement in SAS/ACCESS Interface to Teradata.

```
libname sortprt '.';
proc sort data=sortprt.salaries;
by income;
run;
proc print data=sortprt.salaries;
run;
```

This next example removes the PROC SORT statement shown in the previous example. It instead uses a BY clause with a VAR clause with PROC PRINT. The BY clause returns Teradata rows ordered by the INCOME column.

```
libname sortprt teradata user=myusr1 password=mypwd1;

proc print data=sortprt.salaries;
var income;
by income;
run;
```

## Reducing Workload on Teradata by Sampling

The OBS= option triggers SAS/ACCESS to add a SAMPLE clause to generated SQL. In this example, 10 rows are printed from dbc.ChildrenX:

```
Libname tra teradata user=myusr1 pass=mypwd1 database=dbc;
Proc print data=tra.ChildrenX (obs=10);
run;
```

The SQL passed to Teradata is:

```
SELECT "Child", "Parent" FROM "ChildrenX" SAMPLE 10
```

Especially against large Teradata tables, small values for OBS= reduce workload and spool space consumption on Teradata and your queries complete much sooner. See the SAMPLE clause in your Teradata documentation for more information.

## Security: Authentication with Teradata

### Authentication Using TD2

Teradata 2 (TD2) is the default authentication method that is provided by Teradata. When TD2 is the default authentication method, use your standard user ID and password values. To use TD2 authentication when TD2 is not the default authentication method, append the "@TD2" token to the Teradata user name and provide the corresponding password. Appending the "@TD2" token to a user name overrides any default authentication that is configured.

This example shows how to connect using a TD2 user name:

```
libname x teradata user="janedoe@TD2" password="janedoeworld";
```

For more information about TD2 authentication, see the Teradata security documentation.

## Authentication Using Kerberos

### Authentication and Single Sign-On

Kerberos authentication allows nodes to communicate over a network after providing credentials to the system in a secure manner. Typically, symmetric key encryption is used to mask credentials during the authentication process. Kerberos authentication provides mutual authentication between the nodes on a network. Once authenticated, nodes can communicate freely across the network.

**Note:** Support for Kerberos authentication was added in SAS 9.4M4.

To use single sign-on (SSO) to Teradata with Kerberos authentication, your system must have a valid Kerberos ticket (Ticket Granting Ticket or TGT) before you can initiate a LIBNAME connection to Teradata using Kerberos authentication. On UNIX systems, one way to generate a TGT is to use the `kinit` command. Other methods are available to the security administrator at your site. On the Windows platform, most systems are preconfigured to generate a TGT when you log on and authenticate to your workstation or server. However, methods to authenticate are specific to your site. The security administrator must properly set the TGT expiration time so that any long running jobs have time to complete.

Here are the ways to initiate Kerberos authentication to Teradata in a LIBNAME statement:

- Remove USER= and PASSWORD= options from a LIBNAME statement in existing code, or omit these options when you enter a new LIBNAME statement. This is the recommended way to specify Kerberos single sign-on as your authentication method.
- Indicate SSO by appending your user ID with "@KRB5", such as `USER= "myname@KRB5"`. Alternatively, you can specify simply `USER= "@KRB5"`. This syntax ignores any values that you provide for USER= and PASSWORD= and sets the authentication method to Kerberos (KRB5). Using this syntax overrides any other authentication method that is configured.

**Note:** When using Kerberos authentication, do not also specify the AUTHDOMAIN= LIBNAME option.

Each of the following LIBNAME statements shows how to connect to Teradata using Kerberos authentication:

```
libname teralib1 TERADATA SERVER=xxx ;
libname teralib2 TERADATA SERVER=xxx USER="myname@KRB5"
PASSWORD="myPwd" ;
libname teralib3 TERADATA SERVER=xxx USER="@KRB5" PASSWORD="myPwd" ;
```

```
libname teralib4 TERADATA SERVER=xxx USER="myname@KRB5" ;
libname teralib5 TERADATA SERVER=xxx USER="@KRB5" ;
```

## Common Error Messages

You might encounter the following error messages when attempting Kerberos authentication. Consult this table for guidance.

| Error                                                                           | Meaning                                                                                                                                                                                                |
|---------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERROR: Teradata connection: MTDP: EM_GSSINITFAIL(235): call to gss_init failed. | Kerberos is not enabled on the Teradata client software. See the SAS/ACCESS Interface to Teradata configuration documentation for information about how to enable Kerberos for a SAS client or server. |
| ERROR: Error in the LIBNAME statement.                                          | The Kerberos ticket (TGT) either does not exist, or it is expired. Consult the security administrator at your site.                                                                                    |
| ERROR: Teradata connection: MTDP: EM_GSSCALLFAIL(236): call to gss_call failed. |                                                                                                                                                                                                        |

## Authentication Using LDAP

LDAP is an authentication method that uses an LDAP server to store user names and passwords. Peripheral applications pass user names and passwords to the LDAP server for user authentication. The LDAP authentication system at your organization might also use security realms. These realms specify groups, such as administrators or managers, that have access to specified roles and subsets of information.

For LDAP authentication with either a NULL or single security realm, append only the “@LDAP” token to the Teradata user name. In this case, no realm name is needed. If you append a realm name, the LDAP authentication server ignores it and authentication proceeds.

---

**Note:** To use user names with the “@LDAP” token on Windows, you must first set the GUILOGON environment variable to NO.

---

This example shows how to connect to a single or NULL realm.

```
libname x teradata user="johndoe@LDAP" password="johndoeworld";
```

If your system is configured with multiple security realms, you must append the realm name to the “@LDAP” token. In this case, an LDAP server must already be configured to accept authentication requests from the Teradata server.

Here is an example of how to make the connection to a specific realm, jsrealm, where multiple realms are configured.

```
libname x teradata user="johndoe@LDAPjsrealm" password="johndoeworld";
```

Single sign-on is not supported with LDAP authentication.

## Using the Teradata Wallet Feature

### Overview

The Teradata Wallet feature enables you to store encrypted database passwords for users without exposing those passwords within system scripts. The Teradata Wallet feature is available with TTU 14.00 and higher in UNIX or Windows environments. The Teradata Wallet feature can be used in systems that use LDAP.

Each user's wallet contains items that consist of an item name in plain text and a corresponding encrypted value. The name serves as a hint to identify the encrypted value. For example, a wallet might contain an item with the name *uid\_td2\_server2* and a corresponding value that is the encrypted version of *myuserid*.

Within a script, you call the \$tdwallet function that returns the encrypted value that the system uses in place of a text value. To assign the user ID from the previous example to a parameter that is called *user*, your command might contain this code:

```
... user="$tdwallet(uid_td2_server2)" ...
```

In a SAS program, the values are used similarly.

For more information, including details about adding items to and deleting items from a wallet, see your Teradata documentation.

### Example: Specify Items in a Teradata Wallet

Here is code to specify items in a Teradata wallet on a UNIX command line. The UNIX prompt and command are on the first line. Below this line you can see the resulting prompt and message from the system:

```
machine> tdwallet add uid_td2_server2
Enter desired value for he string named "uid_td2_server2": dbtester
String named "uid_td2_server2" added.
```

The value *dbtester* is stored in an encrypted form in the Teradata wallet, and the encrypted value is passed via the \$tdwallet function as  
"\$tdwallet(uid\_td2\_server2)".

Here are some additional item definitions:

```
machine> tdwallet add password_td2_server2
Enter desired value for he string names "password_td2_server2": db666444333
String named "password_td2_server2" added.
```

```

machine> tdwallet add com.teradata.TD2
Enter desired value for the string named "com.teradata.TD2":
$tdwallet(password_td2_${tdpid})
String named "com.teradata.TD2" added.

machine> tdwallet add uid_ldap_server2
Enter desired value for the string named "uid_ldap_server2": joe_user
String named "uid_ldap_server2" added.

machine> tdwallet add password_ldap_server2
Enter desired value for the string named "password_ldap_server2": joe333444999
String named "password_ldap_server2" added.

machine> tdwallet add com.teradata.LDAP
Enter desired value for the string named "com.teradata.LDAP":
$tdwallet(password_ldap_${tdpid})
String named "com.teradata.LDAP" added.

```

In the item definitions for com.teradata.TD2 and com.teradata.LDAP, the argument to the \$tdwallet function includes \${tdpid}. This is a reference to a predefined Teradata environment variable.

---

## Example: Access Data by Using the Teradata Wallet Feature

Use the following code to access a table that contains data about a new class. The call to the \$tdwallet function with no argument automatically returns the password value for uid\_td2\_server2. All other calls to the \$tdwallet function require arguments.

```

libname td Teradata user="$tdwallet(uid_td2_server2)"
password="$tdwallet"
server=server2;

proc sort data=td.class_new out=lsgout nodupkey;
by age;
run;

```

---

## Example: Access Data in a Teradata Database That Has LDAP and TPT Support

The Teradata Wallet feature works with LDAP and TPT support. The call to the \$tdwallet function with no argument automatically returns the password value for uid\_ldap\_server2. All other calls to the \$tdwallet function require arguments.

```

libname td.teradata user="$tdwallet(uid_ldap_server2)"
password="$tdwallet"
server=server2;

```

```

proc delete data=td.foo;
run;

data td.foo (multiload=yes tpt=yes) ;
  x=55;
run;

```

# Locking in the Teradata Interface

## Overview

The SAS/ACCESS Teradata interface has LIBNAME and data set options that let you modify how Teradata handles locking. Use SAS/ACCESS locking options only when Teradata standard locking is undesirable. For tips on using these options, see “[Understanding SAS/ACCESS Locking Options](#)” on page 1364 and “[When to Use SAS/ACCESS Locking Options](#)” on page 1365. Teradata examples are available.

READ\_LOCK\_TYPE= TABLE | VIEW | ROW  
 UPDATE\_LOCK\_TYPE= TABLE | VIEW  
 READ\_MODE\_WAIT= YES | NO  
 UPDATE\_MODE\_WAIT= YES | NO  
 READ\_ISOLATION\_LEVEL= ACCESS | READ | WRITE  
 Here are the valid values for this option.

*Table 39.3 Read Isolation Levels for Teradata*

| Isolation Level | Definition                                                                                                                                                                                                                                                                                                                                            |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACCESS          | <p>Obtains an ACCESS lock by ignoring other users' ACCESS, READ, and WRITE locks. Permits other users to obtain a lock on the table or view. You see a snapshot of the data when the lock is specified.</p> <p>Can return inconsistent or unusual results if another user is modifying the data at the same time that you are accessing the data.</p> |
| READ            | <p>Obtains a READ lock if no other user holds a WRITE or EXCLUSIVE lock. Does not prevent other users from reading the object.</p> <p>Specify this isolation level whenever possible. READ is usually adequate for most SAS/ACCESS processing.</p>                                                                                                    |
| WRITE           | <p>Obtains a WRITE lock on the table or view if no other user has a READ, WRITE, or EXCLUSIVE lock on the resource. You cannot explicitly release a WRITE lock. It is released</p>                                                                                                                                                                    |

| Isolation Level | Definition                                                                                              |
|-----------------|---------------------------------------------------------------------------------------------------------|
|                 | only when the table is closed. Prevents other users from acquiring any lock but ACCESS.                 |
|                 | This is unnecessarily restrictive, because it locks an entire table until a Read operation is finished. |

`UPDATE_ISOLATION_LEVEL= ACCESS | READ | WRITE`  
 The valid values for this option, ACCESS, READ, and WRITE, are defined in the following table.

*Table 39.4 Update Isolation Levels for Teradata*

| Isolation Level | Definition                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ACCESS          | Obtains an ACCESS lock by ignoring other users' ACCESS, READ, and WRITE locks. Avoids a potential deadlock but can cause data corruption if another user is updating the same data.                                                                                                                                                                                                                                                    |
| READ            | Obtains a READ lock if no other user holds a WRITE or EXCLUSIVE lock. Prevents other users from being granted a WRITE or EXCLUSIVE lock.<br><br>Locks an entire table or view, allowing other users to acquire READ locks. Can lead to deadlock situations.                                                                                                                                                                            |
| WRITE           | Obtains a WRITE lock on the table or view if no other user has a READ, WRITE, or EXCLUSIVE lock on the resource. You cannot explicitly release a WRITE lock. It is released only when the table is closed. Prevents other users from acquiring any lock but ACCESS.<br><br>Prevents all users, except those with ACCESS locks, from accessing a table. Prevents the possibility of a deadlock, but limits concurrent use of the table. |

These locking options cause the LIBNAME engine to transmit a locking request to the DBMS; Teradata performs all data-locking. If you correctly specify a set of SAS/ACCESS read or update locking options, SAS/ACCESS generates locking modifiers that override the Teradata standard locking.

If you specify an incomplete set of locking options, SAS/ACCESS returns an error message. If you do not use SAS/ACCESS locking options, Teradata lock defaults are in effect. For a complete description of Teradata locking, see the LOCKING statement in your Teradata SQL reference documentation.

## Understanding SAS/ACCESS Locking Options

SAS/ACCESS locking options modify Teradata standard locking. Teradata usually locks at the row level. SAS/ACCESS lock options can lock at the row, table, or view level. A change in the scope of a lock from row to table affects concurrent access to

DBMS objects. Specifically, READ and WRITE table locks increase the amount of time that other users must wait to access the table and can decrease overall system performance. These measures help minimize these negative effects.

- Apply READ or WRITE locks *only* when you must apply special locking on Teradata tables.

SAS/ACCESS locking options can be appropriate for special situations, as described in “[When to Use SAS/ACCESS Locking Options](#)” on page 1365. If SAS/ACCESS locking options do not meet your specialized needs, you can use additional Teradata locking features using views. See CREATE VIEW in your Teradata SQL reference documentation for details.

- Limit the span of locks by using data set locking options instead of LIBNAME locking options whenever possible. LIBNAME options affect all tables that you open that your libref references. Data set options apply only to the specified table.

If you specify these read locking options, SAS/ACCESS generates and submits to Teradata locking modifiers that contain the values that you specify for the three read lock options:

- READ\_ISOLATION\_LEVEL= specifies the level of isolation from other table users that is required during SAS/ACCESS Read operations.
- READ\_LOCK\_TYPE= specifies and changes the scope of the Teradata lock during SAS/ACCESS Read operations.
- READ\_MODE\_WAIT= specifies during SAS/ACCESS Read operations whether Teradata should wait to acquire a lock or fail your request when the DBMS resource is locked by another user.

If you specify these update lock options, SAS/ACCESS generates and submits to Teradata locking modifiers that contain the values that you specify for the three update lock options:

- UPDATE\_ISOLATION\_LEVEL= specifies the level of isolation from other table users that is required as SAS/ACCESS reads Teradata rows in preparation for updating the rows.
- UPDATE\_LOCK\_TYPE= specifies and changes the scope of the Teradata lock during SAS/ACCESS Update operations.
- UPDATE\_MODE\_WAIT= specifies during SAS/ACCESS Update operations whether Teradata should wait to acquire a lock or fail your request when the DBMS resource is locked by another user.

## When to Use SAS/ACCESS Locking Options

This section describes situations that might require SAS/ACCESS lock options instead of the standard locking that Teradata provides.

- Use SAS/ACCESS locking options to reduce the isolation level for a Read operation.

When you lock a table using a READ\_\* option, you can lock out both yourself and other users from updating or inserting into the table. Conversely, when other users update or insert into the table, they can lock you out from reading the table. In this situation, you want to reduce the isolation level during a Read

operation. To do this, you specify these read SAS/ACCESS lock options and values.

- READ\_ISOLATION\_LEVEL=ACCESS
- READ\_LOCK\_TYPE=TABLE
- READ\_MODE\_WAIT=YES

One of these situations can result from the options and values in this situation:

- Specify ACCESS locking, eliminating a lock out of yourself and other users. Because ACCESS can return inconsistent results to a table reader, specify ACCESS only if you are casually browsing data, not if you require precise data.
- Change the scope of the lock from row-level to the entire table.
- Request that Teradata wait if it attempts to secure your lock and finds the resource already locked.
- Use SAS/ACCESS lock options to avoid contention.

When you read or update a table, contention can occur: the DBMS waits for other users to release their locks on the table that you want to access. This contention suspends your SAS/ACCESS session. In this situation, to avoid contention during a Read operation, you specify these SAS/ACCESS Read lock options and values.

- READ\_ISOLATION\_LEVEL=READ
- READ\_LOCK\_TYPE=TABLE
- READ\_MODE\_WAIT=NO

One of these situations can result from the options and values in this situation.

- Specify a READ lock.
- Change the scope of the lock. Because you have specified READ\_LOCK\_TYPE=TABLE, when you obtain the lock requested, you lock the entire table until your Read operation finishes.
- Tell SAS/ACCESS to fail the job step if Teradata cannot immediately obtain the READ lock.

## When Lock Options Are Not Recognized

There are situations when the values for locking options are not recognized by Teradata. These are some examples of those situations:

- You attempt to set a lock on a target table that already has a lock in place for the current session. For example, an existing READ lock prevents you from obtaining a lock at the WRITE level.
- You attempt to downgrade the level of a lock that is already in place for a table. For example, you cannot downgrade a ROW lock from READ to ACCESS after the READ lock is in place.
- You specify an aggregate, such as a DISTINCT or GROUP BY operation, in a SELECT statement.

- You have specified READ\_LOCK\_TYPE=ROW, and you do not use the primary or unique secondary index to indicate rows that you want to access. Teradata uses a TABLE lock if a SELECT statement is not limited by a primary or unique secondary index.

To verify the lock type that is used, you can specify MSGLEVEL=1 in the OPTIONS statement, or you can turn on lock logging in Teradata and review the locks in Teradata Viewpoint.

## Examples

### Specifying the Isolation Level to ACCESS for Teradata Tables

```
/* This generates a quick survey of unusual customer purchases. */
libname cust teradata user=myusr1 password=mypwd1
      READ_ISOLATION_LEVEL=ACCESS
      READ_LOCK_TYPE=TABLE
      READ_MODE_WAIT=YES
      CONNECTION=UNIQUE;

proc print data=cust.purchases (where= (bill<2));
run;

data local;
  set cust.purchases (where= (quantity>1000));
run;
```

Here is what SAS/ACCESS does in the above example.

- Connects to the Teradata DBMS and specifies the three SAS/ACCESS LIBNAME Read lock options.
- Opens the Purchases table and obtains an ACCESS lock if another user does not hold an EXCLUSIVE lock on the table.
- Reads and displays table rows with a value less than 2 in the Bill column.
- Closes the Purchases table and releases the ACCESS lock.
- Opens the Purchases table again and obtains an ACCESS lock if a different user does not hold an EXCLUSIVE lock on the table.
- Reads table rows with a value greater than 1000 in the Quantity column.
- Closes the Purchases table and releases the ACCESS lock.

## Setting Isolation Level to WRITE to Update a Teradata Table

```
/* This updates the critical Rebate row. */
libname cust teradata user=myusr1 password=mypwd1;

proc sql;
  update cust.purchases (UPDATE_ISOLATION_LEVEL=WRITE
    UPDATE_MODE_WAIT=YES
    UPDATE_LOCK_TYPE=TABLE)
  set rebate=10 where bill>100;
quit;
```

In this example, here is what SAS/ACCESS does:

- Connects to the Teradata DBMS and specifies the three SAS/ACCESS data set Update lock options.
- Opens the Purchases table and obtains a WRITE lock if a different user does not hold a READ, WRITE, or EXCLUSIVE lock on the table.
- Updates table rows with Bill greater than 100 and sets the Rebate column to 10.
- Closes the Purchases table and releases the WRITE lock.

---

**Note:** UPDATE options are applied only to a preparatory SELECT statement when the SQL statements are passed to Teradata. As a result, deadlock errors are possible if the same table is being updated by other processes at the same time. In this case, the solution is to specify UPDATE\_ISOLATION\_LEVEL=ACCESS.

---

## Preventing a Hung SAS Session When Reading and Inserting to the Same Table

```
/* SAS/ACCESS lock options prevent the session hang */
/* that occurs when reading & inserting into the same table. */
libname tra teradata user=myusr1 password=mypwd1 connection=unique;

proc sql;
  insert into tra.sametable
    select * from tra.sametable(read_isolation_level=access
      read_mode_wait=yes
      read_lock_type=table);
```

Here is what SAS/ACCESS does in the above example:

- Creates a read connection to fetch the rows selected (SELECT \*) from Tra.Sametable and specifies an ACCESS lock (READ\_ISOLATION\_LEVEL=ACCESS). Teradata grants the ACCESS lock.

- Creates an insert connection to Teradata to process the Insert operation to Tra.Sametable. Because the ACCESS lock that is already on the table permits access to the table, Teradata grants a WRITE lock.
- Performs the Insert operation without hanging (suspending) your SAS session.

---

# Naming Conventions for Teradata

---

## Teradata Conventions

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

You can use these conventions to name such Teradata objects as include tables, views, columns, indexes, and macros.

- A name must be from 1 to 32 characters long. Support for 32-character names was added in [SAS 9.4M3](#).

If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name results in identical names, SAS generates a unique name by replacing the last character with a number. If a column name is longer than 32 characters, then SAS/ACCESS attempts to store the full name in the column label. The label must not contain more than 256 bytes. If the label exceeds this 256-byte limit, an error is printed to the SAS log.

DBMS table names must be 32 characters or fewer. SAS does not truncate a longer table name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

---

**Note:** To enable use of long table or column names after upgrading to Teradata 14.10, ensure that you have set the EnableEON DBS control flag to TRUE and then reinitialized the Teradata DBMS.

---

- A name must begin with a letter unless you enclose it in double quotation marks.
- A name can contain letters (A to Z), numbers from 0 to 9, underscore (\_), dollar sign (\$), and the number or pound sign (#). A name in double quotation marks can contain any characters except double quotation marks.
- A name, even when enclosed in double quotation marks, is not case sensitive. For example, CUSTOMER and Customer are the same.
- A name cannot be a Teradata reserved word.
- Because the name must be unique between objects, a view and table in the same database cannot have an identical name.

---

## SAS Naming Conventions

Use these conventions when naming a SAS object:

- A name must be from 1 to 32 characters long.
- A name must begin with a letter (A to Z) or an underscore (\_).
- A name can contain letters (A to Z), numbers from 0 to 9, and an underscore (\_).
- Names are not case sensitive. For example, CUSTOMER and Customer are the same.
- A name cannot be enclosed in double quotation marks.
- A name does not need to be unique between object types.

---

## Naming Objects to Meet Teradata and SAS Conventions

To easily share objects between SAS and the DBMS, create names that meet both SAS and Teradata naming conventions:

- Start with a letter.
- Include only letters, digits, and underscores.
- Use a length of 1 to 32 characters.

---

## Accessing Teradata Objects That Do Not Meet SAS Naming Conventions

---

### Overview

These SAS/ACCESS code examples can help you access Teradata objects (existing Teradata DBMS tables and columns) that have names that do not follow SAS naming conventions.

---

### Example 1: Unusual Teradata Table Name

```
libname unusual teradata user=myusr1 password=mypwd1;
proc sql dquote=ansi;
```

```

create view myview as
  select * from unusual."More names";
  proc print data=myview;run;

```

---

## Example 2: Unusual Teradata Column Names

SAS/ACCESS automatically converts Teradata column names that are not valid for SAS, mapping such characters to underscores. It also appends numeric suffixes to identical names to ensure that column names are unique.

```

create table unusual_names( Name$ char(20), Name# char(20),
                           "Other strange name" char(20))

```

In this example, SAS/ACCESS converts the spaces found in the Teradata column name, OTHER STRANGE NAME, to Other\_strange\_name. After the automatic conversion, SAS programs can then reference the table as usual.

```

libname unusual teradata user=myusr1 password=mypwd1;
proc print data=unusual.unusual_names; run;

```

*Output 39.3 PROC PRINT Display*

| Name_ | Name_0 | Other_strange_name |
|-------|--------|--------------------|
|-------|--------|--------------------|

---

## Data Types for Teradata

---

### Supported Teradata Data Types

Here are the data types that SAS/ACCESS Interface to Teradata supports:

- Binary string data:
  - BYTE(*n*)
  - VARBYTE(*n*)
- Character string data:
  - CHAR (*n*)
  - VARCHAR (*n*)
  - LONG VARCHAR
- Date and time data:
  - DATE

TIME(*n*)

TIMESTAMP (*n*)

Date type columns might contain Teradata values that are out of range for SAS, which handles dates from A.D. 1582 through A.D. 20,000. If SAS/ACCESS encounters an unsupported date (for example, a date earlier than A.D. 1582), it returns an error message and displays the date as a missing value.

---

**Note:** When processing WHERE statements (using PROC SQL or the DATA step) that contain *literal* values for TIME or TIMESTAMP, the SAS engine passes the values to Teradata exactly as they were entered. The values are passed without being rounded or truncated. This is done so that Teradata can handle the rounding or truncation during processing.

---

■ Numeric data:

|                                 |          |
|---------------------------------|----------|
| BYTEINT                         | INTEGER  |
| DECIMAL( <i>n,m</i> )           | NUMBER   |
| FLOAT   REAL   DOUBLE PRECISION | SMALLINT |

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

---

For details about these data types, see your Teradata documentation.

---

## Teradata Null Values

Teradata has a special value that is called NULL. A Teradata NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Teradata NULL value, it interprets it as a SAS missing value.

By default, Teradata columns accept NULL values. However, you can specify columns so that they do not contain NULL values. For example, when you create a SALES table, specify the CUSTOMER column as NOT NULL. This tells Teradata not to add a row to the table unless the CUSTOMER column for the row has a value. When creating a Teradata table with SAS/ACCESS, you can use the DBNULL= data set option to indicate whether NULL is a valid value for specified columns.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how SAS missing character values are handled, use the **NULCHAR=** and **NULCHARVAL=** data set options.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Teradata assigns to SAS variables when using the [LIBNAME statement](#) to read from a Teradata table. SAS/ACCESS does not use Teradata table column attributes when it assigns defaults.

**Table 39.5 Default SAS Formats for Teradata**

| Teradata Data Type                  | Default SAS Format                                            |
|-------------------------------------|---------------------------------------------------------------|
| CHAR( <i>n</i> )                    | \$ <i>n</i> ( <i>n</i> <= 32,767)                             |
| CHAR( <i>n</i> )                    | \$32767.( <i>n</i> >32,767) <sup>1</sup>                      |
| VARCHAR( <i>n</i> )                 | \$ <i>n</i> ( <i>n</i> <= 32,767)                             |
| VARCHAR( <i>n</i> )                 | \$32767.( <i>n</i> > 32,767) <sup>1</sup>                     |
| LONG VARCHAR( <i>n</i> )            | \$32767. <sup>1</sup>                                         |
| BYTE( <i>n</i> )                    | \$HEX <i>n</i> . ( <i>n</i> <= 32,767)                        |
| BYTE( <i>n</i> ) <sup>1</sup>       | \$HEX32767.( <i>n</i> > 32,767)                               |
| VARBYTE( <i>n</i> )                 | \$HEX <i>n</i> . ( <i>n</i> <= 32,767)                        |
| VARBYTE( <i>n</i> )                 | \$HEX32767.( <i>n</i> > 32,767)                               |
| INTEGER                             | 11.0                                                          |
| SMALLINT                            | 6.0                                                           |
| BYTEINT                             | 4.0                                                           |
| DECIMAL( <i>n, m</i> ) <sup>2</sup> | ( <i>n</i> +2 ).( <i>m</i> )                                  |
| FLOAT                               | none                                                          |
| NUMBER <sup>2</sup>                 | <i>w.d</i>                                                    |
| DATE <sup>3</sup>                   | DATE9.                                                        |
| TIME( <i>n</i> ) <sup>4</sup>       | for <i>n</i> =0, TIME8.<br>for <i>n</i> >0, TIME9+ <i>n.n</i> |

| Teradata Data Type                      | Default SAS Format                                       |
|-----------------------------------------|----------------------------------------------------------|
| TIMESTAMP( $n$ ) <sup>4</sup>           | for $n=0$ , DATETIME19.<br>for $n>0$ , DATETIME20+ $n.n$ |
| TRIM(LEADING FROM c)                    | LEFT(c)                                                  |
| CHARACTER_LENGTH(TRIM(TRAILING FROM c)) | LENGTH(c)                                                |
| (v MOD d)                               | MOD(v,d)                                                 |
| TRIMN(c)                                | TRIM(TRAILING FROM c)                                    |

- 1 When reading Teradata data into SAS, DBMS columns that exceed 32,767 bytes are truncated. The maximum size for a SAS character column is 32,767 bytes.
- 2 If the DECIMAL or NUMBER number is extremely large, SAS can lose precision. For details, see “[Supported Teradata Data Types](#)” on page 1371.
- 3 To learn how SAS/ACCESS handles dates that are outside the valid SAS date range, see “[Supported Teradata Data Types](#)” on page 1371.
- 4 TIME and TIMESTAMP are supported for Teradata Version 2, Release 3, and later. The TIME with TIMEZONE, TIMESTAMP with TIMEZONE, and INTERVAL types are presented as SAS character strings and are therefore harder to use.

When you create Teradata tables, the default Teradata columns that SAS/ACCESS creates are based on the type and format of the SAS column. The following table shows the default Teradata data types that SAS/ACCESS assigns to the SAS formats during output processing when you use the LIBNAME statement.

*Table 39.6 Default Output Teradata Data Types*

| SAS Data Type | SAS Format                      | Teradata Data Type |
|---------------|---------------------------------|--------------------|
| Character     | \$w.<br>\$CHARw.<br>\$VARYINGw. | CHAR[w]            |
| Character     | \$HEXw.                         | BYTE[w]            |
| Numeric       | A date format                   | DATE               |
| Numeric       | TIMEw.d                         | TIME(d)            |
| Numeric       | DATETIMEw.d                     | TIMESTAMP(d)       |
| Numeric       | w.(w≤2)                         | BYTEINT            |
| Numeric       | w.(3≤w≤4)                       | SMALLINT           |
| Numeric       | w.(5≤w≤9)                       | INTEGER            |
| Numeric       | w.(≤10w≤18)                     | DECIMAL            |

| SAS Data Type | SAS Format                | Teradata Data Type |
|---------------|---------------------------|--------------------|
| Numeric       | w.(w≥19)                  | NUMBER             |
| Numeric       | w.d                       | DECIMAL(w-1,d)     |
| Numeric       | all other numeric formats | FLOAT              |

To override any default output type, use the DBTYPE= data set option.

---

## Working with NUMBER Data

---

### Working with NUMBER Data in the DATA Step and in Most Procedures

When NUMBER data is read into SAS, SAS/ACCESS automatically converts this data to FLOAT values. Therefore, when you use NUMBER data in a DATA step or in procedures, that data is automatically available for calculations in SAS.

Be aware that when performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that are read are rounded to meet this condition. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see ["Your Options When Choosing Your Needed Degree of Precision" on page 10](#).

The only time when you need to convert NUMBER data to another data type is when you pass native SQL queries to the Teradata database. For more information, see the next section.

---

### Working with NUMBER Data in PROC SQL

When you interact with NUMBER data in native Teradata SQL code that you pass to the database, change the NUMBER data to FLOAT values for use in SAS. To do this, use the Teradata CAST function. This ensures that the data is usable by SAS. Similarly, you can use the CAST function in native Teradata SQL to change from a FLOAT value to a NUMBER value when you write to Teradata.

```
proc sql;
  connect to teradata(user=myuser pass=myPwd
                      server=myServer);
  select * from connection to Teradata(select col1, cast(col2 as
FLOAT), col3
```

```

        from TDlib.table);
... additional code ...
quit;

```

---

## Working with NUMBER Data in PROC FEDSQL and PROC DS2

The FEDSQL and DS2 procedures provide native support for the Teradata NUMBER data type. That is, when you create a Teradata table and define a column of type NUMERIC(p,s) in a FedSQL or DS2 program, SAS creates a column as NUMBER(p,s) in Teradata. There is no need for any conversions to or from the FLOAT data type. For more information about how FedSQL and DS2 data types are mapped to Teradata data types, see “[Data Types for Teradata](#)” in *SAS DS2 Language Reference* or “[Data Types for Teradata](#)” in *SAS DS2 Language Reference*.

```

proc fedsq1;
    create table tera.test(col1 numeric(4,2));
    insert into tera.test values(12.34);
    select * from tera.test;
quit;

proc ds2;
data test;
dcl numeric col1(4,2);
method=run();
    Col1=12.34; output;
end;
endData;
run;
quit;

```

---

## Data Returned as SAS Binary Data with Default Format \$HEX

- BYTE
- VARBYTE
- LONGVARBYTE
- GRAPHIC
- VARGRAPHIC
- LONG VARGRAPHIC

# Temporal Data Types for Teradata

## Overview

Teradata provides built-in support for *temporal data*. Temporal data makes it easy to build and maintain applications where information changes over time. For example, consider a merchandise application that must store the price of an item along with the dates during which that price is valid. As shown below, traditional table design uses two date columns to store the beginning and end dates during which the price is valid.

```
CREATE TABLE price
(
    Item_UPC      BIGINT,
    Price         FLOAT,
    Begin_date   DATE,
    End_date     DATE
)
```

| Item_UPC     | Price | Begin_Date | End_Date   |
|--------------|-------|------------|------------|
| 123456789012 | \$5   | 2010-01-05 | 9999-12-31 |

When the price of the item changes, the end date of the current price must be updated, and a new row must be inserted with the new price.

| Item_UPC     | Price | Begin_Date | End_Date   |
|--------------|-------|------------|------------|
| 123456789012 | \$5   | 2010-01-05 | 2011-05-01 |
| 123456789012 | \$4   | 2011-05-01 | 9999-12-31 |

Teradata temporal support can simplify this process. You can use the PERIOD(DATE) data type to represent the time period during which the price is valid. Here is how you can create the table instead.

```
CREATE TABLE price
(
    Item_UPC      BIGINT,
    Price         FLOAT,
    Price_validity PERIOD(DATE) NOT NULL AS VALIDTIME
)
```

| Item_UPC     | Price | Price_validity           |
|--------------|-------|--------------------------|
| 123456789012 | \$5   | (2010-01-05, 9999-12-31) |

For additional examples and information about features, see the Teradata temporal table support documentation.

---

## Supported Temporal Data Types

SAS/ACCESS Interface to Teradata supports these temporal data types.

- PERIOD(TIME)
- PERIOD(DATE)
- PERIOD(TIMESTAMP)
- PERIOD(DATE) VALIDTIME or TRANSACTIONTIME
- PERIOD(TIMESTAMP) VALIDTIME or TRANSACTIONTIME

---

## Specifying Transaction Time and Valid Time

For true temporal support, you must specify the transaction-time and valid-time attributes on the PERIOD data type.

The transaction-time attribute on a PERIOD column makes the table a transaction-time table. Teradata automatically maintains tables with transaction-time columns. It tracks when a row is first made known to the table. When a row is inserted, it is considered to be an open row because it is currently in effect until the end of time. If the row is deleted, Teradata marks it as a closed row that is no longer in effect. However, the table can be queried to obtain rows that were open at a particular point in time even though the row is not currently valid. Similarly, when a row is modified, the current row is closed and a new row is opened and made effective.

A user cannot specify or modify a transaction-time column.

The valid-time attribute indicates the time period during which the information is in effect. If valid time is specified with the PERIOD data type, Teradata maintains how the time period is in effect if the row is updated or deleted. As in the example about prices, when a row is inserted with a new price, Teradata maintains the end date of the original row. The row with the old price is updated with an end date and the new row is inserted.

A row in a valid-time transaction table can be a history row, a current row, or a future row. The history row is no longer valid with respect to current time. Its end-time period is before the current time. A current row has a time period that straddles the current time.

---

## Creating a Table from SAS with the PERIOD Data Type

To create a Teradata table with temporal data types from SAS, use the DBTYPE= data set option. SAS does not have an equivalent data type for PERIOD. The value is represented in SAS as a character string. In this example, when the character

string that represents the period ID is inserted into Teradata, it is implicitly converted to a PERIOD data type.

```
data x.mytest(DBTYPE=(validity='PERIOD(DATE) VALIDTIME'));
      i=1;
      validity='(1973-02-03, 9999-12-31)';
      output;
run;
```

## Reading in a PERIOD Data Type

A Teradata PERIOD data-type column can be read into SAS like any other column. It is represented in SAS as a character string, such as '(1973-02-03, 9999-12-31)'.

## Temporal Qualifiers

Temporal tables contain rows that can be current, history, or future in the valid-time dimension. In the transaction-time dimension, rows can be open or closed. Temporal qualifiers specify what data is needed. The TEMPORAL\_QUALIFIER=LIBNAME and data set options let you qualify queries in the valid-time or transaction-time dimension. For example, to fetch rows that are valid as of '2009-01-01' in a table, you must specify TEMPORAL\_QUALIFIER='VALIDTIME AS OF DATE '2009-01-01'' as a LIBNAME or data set option when you query temporal tables.

The option that you specify for TEMPORAL\_QUALIFIER= is free-form text. Here are some examples.

```
TEMPORAL_QUALIFIER='CURRENT VALIDTIME'
TEMPORAL_QUALIFIER='VALIDTIME AS OF DATE '2009-01-01' '
TEMPORAL_QUALIFIER='NONSEQUENCED VALIDTIME'
TEMPORAL_QUALIFIER='    SEQUENCED VALIDTIME'
TEMPORAL_QUALIFIER='NONSEQUENCED VALIDTIME PERIOD '(2007-01-01, 2008-03-01)'''
TEMPORAL_QUALIFIER='    SEQUENCED VALIDTIME PERIOD '(2007-01-01, 2008-03-01)'''
TEMPORAL_QUALIFIER='CURRENT TRANSACTIONTIME'
TEMPORAL_QUALIFIER='TRANSACTIONTIME AS OF TIMESTAMP '2009-01-01 01:02:03.123456' ''
```

If you specify the temporal qualifier on the LIBNAME, it applies to the entire session because it is implemented by issuing session commands at connect time. For example, if you specify TEMPORAL\_QUALIFIER='ASOF PERIOD '(1999-01-01, 2099-01-05)' ' on the LIBNAME, here is the Teradata SET SESSION command that is issued at connect time. The SQL is submitted as usual.

```
.SET SESSION ASOF PERIOD '(1999-01-01, 2099-01-05)'
```

If you submit the above command, the temporal qualifier is added as a prefix, as shown below.

```
ASOF PERIOD '(1999-01-01, 2099-01-05)'
SELECT * from TEMPORAL_TABLE;
```

Sample code

-----

```

/* PERIOD data types require the Teradata V13 server. */
libname x teradata user=mysr1 pw=mypwd1 server=mysr1;

/* Create a table with the PERIOD(DATE) data type.
   Note: This is not a temporal table. */
data x.mytest(DBTYPE=(validity='PERIOD(DATE)' ));
i=1; validity='(1973-02-03, 9999-12-31)'; output;
run;

/* Read from a table with a PERIOD data type? */
proc print data=x.mytest;
run;

/* Use Fastload to load a table with a PERIOD data type. */
proc datasets library=x;
  delete mytest;run;

data x.mytest(DBTYPE=(validity='PERIOD(TIMESTAMP)' ) FASTLOAD=YES
TPT=NO);
i=1; validity='(1970-01-05 01:02:03.123, 1970-01-05 05:06:07.456)';
output;
run;

/* Temporal support starts in Teradata V13.10. */
libname x teradata user=mysr1 pw=mypwd1 server=mysr1;

/* Create a table with the PERIOD(DATE) data type. */
data x.mytest(DBTYPE=(validity='PERIOD(DATE) VALIDTIME'));
i=1; validity='(1973-02-03, 1999-12-31)'; output;
i=2; validity='(2000-01-01, 2011-01-01)'; output;
i=3; validity='(2011-01-02, 9999-12-31)'; output;
run;

/* Can we read a PERIOD data type?
   You must select the row with i=2. */
proc print data=x.mytest(TEMPORAL_QUALIFIER='CURRENT VALIDTIME');
run;

/* Consider data as of 1995-01-01. */
libname x teradata user=mysr1 pw=mypwd1 server=mysr1
TEMPORAL_QUALIFIER='VALIDTIME AS OF DATE '1995-01-01'';

/* Row with i=1 is returned. */
proc print data=x.mytest(DBSLICEPARAM=ALL);
run;

```

---

## Sample Programs for Teradata

You can view sample programs for this SAS/ACCESS interface in the [SAS Software GitHub](#) repository.

You can find the samples for Base SAS under SAS Foundation. Samples that run for all engines are at the top level. Samples that are specific to your interface can be found in the folder for your interface. For additional information about using the sample files, see the GitHub repository.



# SAS/ACCESS Interface to Vertica

---

|                                                                      |      |
|----------------------------------------------------------------------|------|
| <i>System Requirements for SAS/ACCESS Interface to Vertica</i> ..... | 1383 |
| <i>Introduction to SAS/ACCESS Interface to Vertica</i> .....         | 1384 |
| <b><i>LIBNAME Statement for the Vertica Engine</i></b> .....         | 1384 |
| Overview .....                                                       | 1384 |
| Arguments .....                                                      | 1385 |
| Vertica LIBNAME Statement Examples .....                             | 1389 |
| <b><i>Data Set Options for Vertica</i></b> .....                     | 1389 |
| <b><i>SQL Pass-Through Facility Specifics for Vertica</i></b> .....  | 1391 |
| Key Information .....                                                | 1391 |
| CONNECT Statement Examples .....                                     | 1392 |
| <b><i>Understanding Vertica Update and Delete Rules</i></b> .....    | 1392 |
| <b><i>Temporary Table Support for Vertica</i></b> .....              | 1393 |
| <b><i>Passing SAS Functions to Vertica</i></b> .....                 | 1394 |
| <b><i>Passing Joins to Vertica</i></b> .....                         | 1395 |
| <b><i>Locking in the Vertica Interface</i></b> .....                 | 1395 |
| <b><i>Naming Conventions for Vertica</i></b> .....                   | 1396 |
| <b><i>Data Types for Vertica</i></b> .....                           | 1397 |
| Overview .....                                                       | 1397 |
| Supported Vertica Data Types .....                                   | 1397 |
| Vertica Null Values .....                                            | 1398 |

---

## System Requirements for SAS/ACCESS Interface to Vertica

You can find information about system requirements for SAS/ACCESS Interface to Vertica in the following locations:

- [System Requirements for SAS/ACCESS Interface to Vertica with SAS 9.4](#)
  - [SAS Viya System Requirements](#)
  - [Third-Party Software Requirements](#)
- 

## Introduction to SAS/ACCESS Interface to Vertica

For available SAS/ACCESS features, see [Vertica supported features on page 119](#). For more information about Vertica, see your Vertica documentation.

Starting in SAS Viya 3.4, SAS/ACCESS Interface to Vertica includes SAS Data Connector to Vertica. The data connector enables you to transfer large amounts of data between your Vertica database and the CAS server for parallel processing. For more information see these sections:

- [“Where to Specify Data Connector Options” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
  - [“Vertica Data Connector” in \*SAS Cloud Analytic Services: User’s Guide\*](#)
- 

## LIBNAME Statement for the Vertica Engine

### Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Vertica supports. For general information about this feature, see [LIBNAME Statement for Relational Databases on page 127](#).

Here is the LIBNAME statement syntax for accessing Vertica.

**LIBNAME libref vertica <connection-options> <LIBNAME-options>;**

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see [“LIBNAME Statement” in \*SAS Global Statements: Reference\*](#).

---

## Arguments

*libref*

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

*vertica*

specifies the SAS/ACCESS engine name for the Vertica interface.

*connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the Vertica server in one of two ways. Specify only one of these methods for each connection because they are mutually exclusive.

- SERVER=, DATABASE=, PORT=, USER=, PASSWORD=
- DSN=, USER=, PASSWORD=

Here is how these options are defined.

---

**Note:** All of the following connection options are also valid in the CONNECT statement when you use the SQL pass-through facility (SQL procedure) to connect to your DBMS.

---

**SERVER=<'>*Vertica-server-name*<'>**

specifies the server name or IP address of the Vertica server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**DATABASE=<'>*Vertica-database-name*<'>**

specifies the name of the database on the Vertica server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORt=port**

specifies the port number that is used to connect to the specified Vertica server. If you do not specify a port, the default is 5433.

**USER=<'>*Vertica-user-name*<'>**

specifies the Vertica user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: UID

**PASSWORD=<'>*Vertica-password*<'>**

specifies the password that is associated with your Vertica user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: PWD=

**DSN=<'>Vertica-data-source<'>**

specifies the configured Vertica ODBC data source to which you want to connect. Use this option if you have existing Vertica ODBC data sources that are configured on your client. This method requires additional setup. This extra setup is done through the ODBC Administrator control panel on Windows platforms, through the odbc.ini file, or a similarly named configuration file on UNIX platforms. Therefore, it is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

**PRESERVE\_COMMENTS**

lets you pass additional information (called hints) to Vertica DBMS for processing. These hints might direct the Vertica DBMS query optimizer to choose the best processing method based on your hint.

You specify PRESERVE\_COMMENTS as an argument in the CONNECT statement. You then specify the hints in the Vertica DBMS SQL query for the CONNECTION TO component. Hints are entered as comments in the SQL query and are passed to and processed by Vertica DBMS.

***LIBNAME-options***

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance; others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Vertica, with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

*Table 40.1 SAS/ACCESS LIBNAME Options for Vertica*

| Option              | Default Value                                                                        | Valid in CONNECT |
|---------------------|--------------------------------------------------------------------------------------|------------------|
| ACCESS=             | none                                                                                 |                  |
| AUTHDOMAIN=         | none                                                                                 |                  |
| AUTOCOMMIT=         | NO                                                                                   | •                |
| CONNECTION=         | UNIQUE                                                                               | •                |
| CONNECTION_GROUP=   | none                                                                                 | •                |
| CONOPTS=            | none                                                                                 | •                |
| DBCLIENT_MAX_BYTES= | matches the maximum number of bytes per single character of the SAS session encoding | •                |
| DBCOMMIT=           | 1000 when inserting rows,<br>0 when updating rows                                    |                  |
| DBCONINIT=          | none                                                                                 | •                |

| Option                    | Default Value       | Valid in CONNECT |
|---------------------------|---------------------|------------------|
| DBCONTERM=                | none                | •                |
| DBCREATE_TABLE_OPTS=      | none                |                  |
| DBGEN_NAME=               | DBMS                | •                |
| DBINDEX=                  | NO                  |                  |
| DBLIBINIT=                | none                |                  |
| DBLIBTERM=                | none                |                  |
| DBMAX_TEXT=               | 1024                | •                |
| DBMSTEMP=                 | NO                  |                  |
| DBNULLKEYS=               | YES                 |                  |
| DBPROMPT=                 | NO                  | •                |
| DBSASLABEL=               | COMPAT              |                  |
| DBSERVER_MAX_BYTES=       | usually 1           | •                |
| DBSLICEPARM=              | none                |                  |
| DEFER=                    | NO                  | •                |
| DELETE_MULT_ROWS=         | NO                  |                  |
| DIRECT_EXE=               | none                |                  |
| DIRECT_SQL=               | YES                 |                  |
| IGNORE_READ_ONLY_COLUMNS= | NO                  |                  |
| INSERT_SQL=               | YES                 |                  |
| INSERTBUFF=               | based on row length |                  |
| KEYSET_SIZE=              | 0                   |                  |
| MULTI_DATASRC_OPT=        | NONE                |                  |
| POST_STMT_OPTS=           | none                |                  |

| Option                  | Default Value                               | Valid in CONNECT |
|-------------------------|---------------------------------------------|------------------|
| PRESERVE_COL_NAMES=     | see “Data Types for Vertica”                |                  |
| PRESERVE_TAB_NAMES=     | see “Data Types for Vertica”                |                  |
| QUALIFIER=              | none                                        |                  |
| QUERY_TIMEOUT=          | 0                                           | •                |
| QUOTE_CHAR=             | none                                        |                  |
| READ_ISOLATION_LEVEL=   | RC (see “Locking in the Vertica Interface”) | •                |
| READ_LOCK_TYPE=         | ROW                                         | •                |
| READBUFF=               | 1                                           | •                |
| REREAD_EXPOSURE=        | NO                                          | •                |
| SCHEMA=                 | none                                        |                  |
| SPOOL=                  | YES                                         |                  |
| SQL_FUNCTIONS=          | none                                        |                  |
| SQL_FUNCTIONS_COPY=     | none                                        |                  |
| SQLGENERATION=          | none                                        |                  |
| STRINGDATES=            | NO                                          | •                |
| SUB_CHAR=               | none                                        |                  |
| TRACE=                  | NO                                          | •                |
| TRACEFILE=              | none                                        | •                |
| UPDATE_ISOLATION_LEVEL= | RC (see “Locking in the Vertica Interface”) | •                |
| UPDATE_LOCK_TYPE=       | ROW                                         | •                |
| UPDATE_MULT_ROWS=       | NO                                          |                  |
| UPDATE_SQL=             | YES                                         |                  |

| Option              | Default Value | Valid in CONNECT |
|---------------------|---------------|------------------|
| UTILCONN_TRANSIENT= | NO            |                  |

## Vertica LIBNAME Statement Examples

No DSN is specified in this example. This example uses the recommended default values for the connection options to make the connection.

```
libname mydblib vertica server="mysrv1" port=5433
      user=myusr1 password=mypwd1 database=mydb1;
```

A DSN is specified in this next example.

```
libname mydblib vertica dsn=mydsn1
      user=myusr1 password=mypwd1;
```

## Data Set Options for Vertica

All SAS/ACCESS data set options in this table are supported for Vertica. Default values are provided where applicable. For details, see [Data Set Options for Relational Databases on page 370](#).

*Table 40.2 SAS/ACCESS Data Set Options for Vertica*

| Option               | Default Value        |
|----------------------|----------------------|
| DBCOMMIT=            | LIBNAME option value |
| DBCONDITION=         | none                 |
| DBCREATE_TABLE_OPTS= | LIBNAME option value |
| DBFORCE=             | NO                   |
| DBGEN_NAME=          | DBMS                 |
| DBINDEX=             | LIBNAME option value |
| DBKEY=               | none                 |
| DBLABEL=             | NO                   |

| Option                    | Default Value                                    |
|---------------------------|--------------------------------------------------|
| DBLARGETABLE=             | none                                             |
| DBMAX_TEXT=               | 1024                                             |
| DBNULL=                   | YES                                              |
| DBNULLKEYS=               | LIBNAME option value                             |
| DBPROMPT=                 | LIBNAME option value                             |
| DBSASLABEL=               | COMPAT                                           |
| DBSASTYPE=                | see “Data Types for Vertica” on page 1397        |
| DBSLICE=                  | none                                             |
| DBSLICEPARM=              | NONE                                             |
| DBTYPE=                   | none (see “Data Types for Vertica” on page 1397) |
| ERRLIMIT=                 | 1                                                |
| IGNORE_READ_ONLY_COLUMNS= | NO                                               |
| INSERT_SQL=               | LIBNAME option value                             |
| INSERTBUFF=               | LIBNAME option value                             |
| NULLCHAR=                 | SAS                                              |
| NULLCHARVAL=              | a blank character                                |
| POST_STMT_OPTS=           | none                                             |
| POST_TABLE_OPTS=          | none                                             |
| PRE_STMT_OPTS=            | none                                             |
| PRE_TABLE_OPTS=           | none                                             |
| PRESERVE_COL_NAMES=       | LIBNAME option value                             |
| QUALIFIER=                | LIBNAME option value                             |
| QUERY_TIMEOUT=            | LIBNAME option value                             |

| Option                  | Default Value                                                              |
|-------------------------|----------------------------------------------------------------------------|
| READ_ISOLATION_LEVEL=   | LIBNAME option value (see “Locking in the Vertica Interface” on page 1395) |
| READ_LOCK_TYPE=         | LIBNAME option value                                                       |
| READBUFF=               | LIBNAME option value                                                       |
| SASDATEFMT=             | none                                                                       |
| SCHEMA=                 | LIBNAME option value                                                       |
| SUB_CHAR=               | none                                                                       |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value (see “Locking in the Vertica Interface” on page 1395) |
| UPDATE_LOCK_TYPE=       | LIBNAME option value                                                       |
| UPDATE_SQL=             | LIBNAME option value                                                       |

## SQL Pass-Through Facility Specifics for Vertica

### Key Information

For general information about this feature, see “SQL Pass-Through Facility” on page 690.

Here are the SQL pass-through facility specifics for the Vertica interface.

- The *dbms-name* is VERTICA.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Vertica. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default VERTICA alias is used.
- The *database-connection-arguments* for the CONNECT statement are identical to its LIBNAME [connection options](#).

## CONNECT Statement Examples

This example uses the DBCON alias to connect to the `mydb1` Vertica database, execute several queries, and then disconnect. The connection alias is optional.

```
proc sql;
  connect to vertica as dbcon
    (server=mysrv1 port=5433 user=myusr1 password=mypwd1 db=mydb1);

  execute (drop table scnpastbl) by dbcon;
  execute (create table scnpastbl (id int,
    name char(13), tel char(11), constraint_vpk
    primary key(id, tel))) by dbcon;
  execute (insert into scnpastbl values
    (1, '111', '1-1-1')) by dbcon;

  select * from connection to dbcon
    (select * from scnpastbl);

  disconnect from dbcon;
quit;
```

In this example, the `PRESERVE_COMMENTS` argument is specified after the `USER=` and `PASSWORD=` arguments. The Vertica DBMS SQL query is enclosed in the required parentheses.

```
Connect to vertica DBMS as mycon(user=myusr1
password=mypwd1 preserve_comments);
select *
from connection to mycon
(select /* +indx(empid) all_rows */
count(*) from employees);
quit;
```

Hints are not preserved in this next example, which uses an older style of syntax.

```
execute ( delete /*+ FIRST_ROWS */ from test2 where num2=1) by &db
```

Using the new syntax, hints are preserved in this example.

```
execute by &db ( delete /*+ FIRST_ROWS */ from test2 where num2=2);
```

## Understanding Vertica Update and Delete Rules

To avoid data integrity problems when updating or deleting data, you must specify a primary key on your table.

This example uses `DBTYPE=` to create the primary key.

```

libname invty vertica server=mysrv1 port=5433 user=myusr1 database=mydb1;

proc sql;
drop table invty STOCK23;
quit;

data invty STOCK23 (DBTYPE=(RECDATE="date not null,primary key(RECDATE)")) ;
  input PARTNO $ DESCX $ INSTOCK @17
        RECDATE date7. @25 PRICE;
  format RECDATE date7. ;
  datalines;

K89R  seal      34  27jul95  245.00
M447  sander    98  20jun95  45.88
LK43   filter    121 19may96  10.99
MN21   brace     43  10aug96  27.87
BC85   clamp     80  16aug96   9.55
KJ66   cutter     6  20mar96  24.50
UYN7   rod       211 18jun96  19.77
JD03   switch    383 09jan97  13.99
BV1I   timer     26  03jan97  34.50
;

```

These next examples show how you can update the table now that STOCK23 has a primary key.

```

proc sql;
update invty STOCK23 set price=price*1.1 where INSTOCK > 50;
quit;

proc sql;
delete from invty STOCK23 where INSTOCK > 150;
quit;

```

---

**Note:** Vertical does not enforce uniqueness of primary keys when they are loaded into a table. Instead, consider using sequences or auto-incrementing of columns for primary-key columns. It guarantees uniqueness and avoids the problem of constraint enforcement and the associated overhead. For more information about sequencing, see your Vertical documentation.

---

## Temporary Table Support for Vertical

SAS/ACCESS Interface to Vertical supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page 51.

# Passing SAS Functions to Vertica

SAS/ACCESS Interface to Vertica passes the following SAS functions to Vertica for processing. Where the Vertica function name differs from the SAS function name, the Vertica name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                            |                   |
|----------------------------|-------------------|
| ABS                        | MAX               |
| ARCOS (ACOS)               | MIN               |
| ARSIN (ASIN)               | MINUTE            |
| ATAN                       | MONTH             |
| ATAN2                      | QTR (QUARTER)     |
| Avg                        | REPEAT            |
| BYTE                       | SECOND            |
| CEIL                       | SIGN              |
| COALESCE                   | SIN               |
| COS                        | SQRT              |
| EXP                        | STD (STDDEV_SAMP) |
| FLOOR                      | STRIP (BTRIM)     |
| HOUR                       | SUBSTR            |
| INDEX (STRPOS)             | TAN               |
| LENGTH                     | TRANWRD (REPLACE) |
| LENGTHC (CHARACTER_LENGTH) | TRIMN (RTRIM)     |
| LENGTHN (LENGTH)           | UPCASE (UPPER)    |
| LOG (LN)                   | VAR (VAR_SAMP)    |
| LOG10 (LOG)                | YEAR              |
| LOWCASE (LOWER)            |                   |

`SQL_FUNCTIONS=ALL` allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when `SQL_FUNCTIONS=ALL` can the SAS/ACCESS engine also pass these SAS SQL functions to Vertica. Due to incompatibility in date and time functions between Vertica and SAS, Vertica might not process them correctly. Check your results to determine whether these functions are working as expected.

|                      |                          |
|----------------------|--------------------------|
| COMPRESS (TRANSLATE) | DTEXTDAY (DAYOFMONTH)    |
| DATE (CURRENT_DATE)  | DTEXTWEEKDAY (DAYOFWEEK) |
| DATEPART (DATE)      | TIME (CLOCK_TIMESTAMP)   |
| DATETIME (GETDATE)   | TIMEPART (CAST)          |
| DAY (DAYOFMONTH)     | TODAY (CURRENT_DATE)     |
| DTEXTYEAR (YEAR)     | TRANSLATE                |
| DTEXTMONTH (MONTH)   | WEEKDAY (DAYOFWEEK)      |

---

## Passing Joins to Vertica

For a multiple libref join to pass to Vertica, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- port (PORT=)
- data source (DSN=, if specified)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see “[Passing Joins to the DBMS](#)” on page 61.

---

## Locking in the Vertica Interface

These LIBNAME and data set options let you control how the Vertica interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134.

`READ_LOCK_TYPE=ROW`

`UPDATE_LOCK_TYPE=ROW`

`READ_ISOLATION_LEVEL=S | RR | RC | RU | V`

The ODBC driver manager supports the S, RR, RC, RU, and V isolation levels that are defined in this table.

*Table 40.3 Isolation Levels for Vertica*

| Isolation Level       | Definition                                                                   |
|-----------------------|------------------------------------------------------------------------------|
| S (serializable)      | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.           |
| RR (repeatable read)  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads. |
| RC (read committed)   |                                                                              |
| RU (read uncommitted) | Allows dirty Reads, nonrepeatable Reads, and phantom Reads,                  |

| Isolation Level | Definition                                                                                                                                                                                                                      |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| V (versioning)  | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads. These transactions are serializable but higher concurrency is possible than with the serializable isolation level. Typically, a nonlocking protocol is used. |

Here are how the terms in the table are defined.

*Dirty reads*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it can see changes that are made by those concurrent transactions even before they commit.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

*Phantom reads*

When a transaction exhibits this phenomenon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist, a phantom.

`UPDATE_ISOLATION_LEVEL=S | RR | RC | V`

The ODBC driver manager supports the S, RR, RC, and V isolation levels defined in the preceding table.

---

## Naming Conventions for Vertica

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Vertica interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Vertica is not case sensitive, and all names default to lowercase.

Vertica objects include tables, views, and columns. Follow these naming conventions:

- A name must be from 1 to 128 characters long.
- A name must begin with a letter (A through Z), diacritic marks, or non-Latin characters (200-377 octal)
- A name cannot begin with an underscore (\_). Leading underscores are reserved for system objects.
- Names are not case sensitive. For example, CUSTOMER and Customer are the same, but object names are converted to lowercase when they are stored in the Vertica database. However, if you enclose a name in quotation marks, it is case sensitive.
- A name cannot be a Vertica reserved word, such as WHERE or VIEW.
- A name cannot be the same as another Vertica object that has the same type.

# Data Types for Vertica

## Overview

Every column in a table has a name and a data type. The data type tells Vertica how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Vertica data types, null and default values, and data conversions.

For more information about Vertica data types and to determine which data types are available for your version of Vertica, see your Vertica documentation.

SAS/ACCESS Interface to Vertica does not directly support any data types that are not listed below. Any columns using these types are read into SAS as character strings.

## Supported Vertica Data Types

Here are the data types that the Vertica engine supports.

- Binary string data: BINARY
- String data:
  - CHAR    VARCHAR
- Numeric data:
 

|                  |         |
|------------------|---------|
| BOOLEAN          | INT8    |
| DOUBLE PRECISION | SMALINT |
| FLOAT            | TINYINT |

|               |         |
|---------------|---------|
| FLOAT(n)      | DECIMAL |
| FLOAT8        | NUMERIC |
| REAL          | NUMBER  |
| INTEGER   INT | MONEY   |
| BIGINT        |         |

---

**Note:** When performing calculations on numeric values and when storing numeric values, SAS maintains up to 15 digits of precision. When you read values that contain more than 15 decimal digits of precision from a database into SAS, the values that SAS reads are rounded to meet this condition. When you use a large numeric value in a WHERE clause, this rounding can cause unexpected results, such as not selecting desired rows. For noncomputational purposes, such as storing ID values or credit card numbers, you can read the data in as character data. For more information, see “[Your Options When Choosing Your Needed Degree of Precision](#)” on page 10.

---

- Date, time, and timestamp data:

|               |                         |
|---------------|-------------------------|
| DATE          | TIME WITH TIMEZONE      |
| DATETIME      | TIMESTAMP               |
| SMALLDATETIME | TIMESTAMP WITH TIMEZONE |
| TIME          | INTERVAL                |

---

## Vertica Null Values

Many relational database management systems have a special value called NULL. A DBMS NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a DBMS NULL value, it interprets it as a SAS missing value.

In most relational databases, columns can be specified as NOT NULL so that they require data (they cannot contain NULL values). When a column is specified as NOT NULL, the DBMS does not add a row to the table unless the row has a value for that column. When creating a DBMS table with SAS/ACCESS, you can use the **DBNULL=** data set option to indicate whether NULL is a valid value for specified columns.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how the DBMS handles SAS missing character values, use the **NULCHAR=** and **NULCHARVAL=** data set options.

# SAS/ACCESS Interface to Yellowbrick

|                                                                          |             |
|--------------------------------------------------------------------------|-------------|
| <i>System Requirements for SAS/ACCESS Interface to Yellowbrick</i> ..... | <b>1400</b> |
| <i>Introduction to SAS/ACCESS Interface to Yellowbrick</i> .....         | <b>1400</b> |
| <b><i>LIBNAME Statement for the Yellowbrick Engine</i></b> .....         | <b>1400</b> |
| Overview .....                                                           | 1400        |
| Arguments .....                                                          | 1401        |
| LIBNAME Statement Examples for Yellowbrick .....                         | 1405        |
| <b><i>Data Set Options for Yellowbrick</i></b> .....                     | <b>1405</b> |
| <b><i>Known Limitation When Working with Yellowbrick</i></b> .....       | <b>1407</b> |
| <b><i>SQL Pass-Through Facility Specifics for Yellowbrick</i></b> .....  | <b>1408</b> |
| Key Information .....                                                    | 1408        |
| CONNECT Statement Example for Yellowbrick .....                          | 1408        |
| <b><i>Temporary Table Support for Yellowbrick</i></b> .....              | <b>1408</b> |
| <b><i>Passing SAS Functions to Yellowbrick</i></b> .....                 | <b>1409</b> |
| <b><i>Passing Joins to Yellowbrick</i></b> .....                         | <b>1410</b> |
| <b><i>Bulk Loading and Unloading for Yellowbrick</i></b> .....           | <b>1410</b> |
| Overview of Bulk Loading and Unloading for Yellowbrick .....             | 1410        |
| LIBNAME and Data Set Options with Bulk Loading and Unloading .....       | 1411        |
| Bulk-Load Example for Yellowbrick .....                                  | 1411        |
| <b><i>Locking in the Yellowbrick Interface</i></b> .....                 | <b>1412</b> |
| <b><i>Naming Conventions for Yellowbrick</i></b> .....                   | <b>1413</b> |
| <b><i>Data Types for Yellowbrick</i></b> .....                           | <b>1414</b> |
| Overview of Data Types .....                                             | 1414        |
| Yellowbrick Null Values .....                                            | 1414        |
| Working with Long Character Values in Yellowbrick .....                  | 1415        |
| LIBNAME Statement Data Conversions .....                                 | 1416        |
| Items of Note for Yellowbrick Data .....                                 | 1417        |

---

# System Requirements for SAS/ACCESS Interface to Yellowbrick

You can find information about system requirements for SAS/ACCESS Interface to Yellowbrick in the following locations:

- [SAS Viya System Requirements](#)
- [Third-Party Software Requirements](#)

---

# Introduction to SAS/ACCESS Interface to Yellowbrick

For available SAS/ACCESS features, see “[SAS/ACCESS Interface to Yellowbrick: Supported Features](#)” on page 120. For more information about Yellowbrick, see your Yellowbrick documentation.

---

# LIBNAME Statement for the Yellowbrick Engine

---

## Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Yellowbrick supports. For general information about this feature, see [LIBNAME Statement for Relational Databases](#) on page 127.

Here is the LIBNAME statement syntax for accessing Yellowbrick.

**LIBNAME** *libref* **ybrick** <connection-options> <LIBNAME-options>;

For general information about the LIBNAME statement that is not specific to SAS/ACCESS, see “[LIBNAME Statement](#)” in [SAS Global Statements: Reference](#).

---

## Arguments

***libref***

specifies any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables and views.

***ybrick***

specifies the SAS/ACCESS engine name for the Yellowbrick interface.

***connection-options***

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. Here is how these options are specified.

There are two methods to connect to a Yellowbrick database. Choose one group of options or the other. Additional connection options are optional.

- SERVER=, DATABASE=, PORT=, USER=, and PASSWORD=
- DSN=, USER=, and PASSWORD=

---

**Note:** All of these connection options (preceding the table of Yellowbrick LIBNAME statement options) are valid when used in the CONNECT statement with the SQL procedure.

---

**SERVER=<'>*Yellowbrick-server-name*<'>**

specifies the server name or IP address of the Yellowbrick server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**DATABASE=<'>*Yellowbrick-database-name*<'>**

specifies the name of the database on the Yellowbrick server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: DB=

**PORT=*port***

specifies the port number that is used to connect to the specified Yellowbrick server.

Default: 5432

**USER=<'>*Yellowbrick-user-name*<'>**

specifies the Yellowbrick user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: UID=

**PASSWORD=<'>*Yellowbrick-password*<'>**

specifies the password that is associated with your Yellowbrick user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

Alias: PASS=, PW=, PWD=

**DSN=<'>Yellowbrick-data-source<'>**

specifies the configured Yellowbrick ODBC data source to which you want to connect. Use this option if you have existing Yellowbrick ODBC data sources that are configured on your client. This method requires additional setup—either through the Administrator control panel on Windows platforms or through the odbc.ini file or a similarly named configuration file on UNIX platforms. So it is recommended that you use this connection method only if you have existing, functioning data sources that have been specified.

**COMPLETE=<'>CLI-connection-string<'>**

specifies connection information for your database for PCs only. Separate multiple options with a semicolon. When a successful connection is made, the complete connection string is returned in the SYSDBMSG macro variable. If you do not specify enough correct connection options, you are prompted with a dialog box that displays the values from the COMPLETE= connection string. You can edit any field before you connect to the database. See your Yellowbrick documentation for more details.

This option is not available on UNIX platforms.

**PROMPT=<'>Yellowbrick-connection-options<'>**

specifies connection options for your database. Separate multiple options with a semicolon. When connection succeeds, the complete connection string is returned in the SYSDBMSG macro variable. PROMPT= does not immediately try to connect to the DBMS. Instead, it displays a dialog box in SAS Display Manager that contains the values that you entered in the PROMPT= connection string. You can edit values or enter additional values in any field before you connect to the database.

Restriction: This option is not available on UNIX platforms.

**NOPROMPT=<'>Yellowbrick-connection-options<'>**

specifies connection options for your database. Separate multiple options with a semicolon. If you do not specify enough correct connection options, an error is returned. No dialog box is displayed to help you with the connection string.

***LIBNAME-options***

specify how SAS processes DBMS objects. Some LIBNAME options can enhance performance, and others determine locking or naming behavior. The following table describes the LIBNAME options for SAS/ACCESS Interface to Yellowbrick with the applicable default values. This table also identifies LIBNAME options that are valid in the CONNECT statement in the SQL procedure. For details, see [LIBNAME Options for Relational Databases on page 134](#).

**Table 41.1** SAS/ACCESS LIBNAME Options for Yellowbrick

| Option      | Default Value | Valid in CONNECT |
|-------------|---------------|------------------|
| ACCESS=     | none          |                  |
| AUTHDOMAIN= | none          | •                |
| AUTOCOMMIT= | NO            | •                |
| BULKLOAD=   | NO            | •                |

| Option               | Default Value                                                                        | Valid in CONNECT |
|----------------------|--------------------------------------------------------------------------------------|------------------|
| BULKUNLOAD=          | NO                                                                                   | •                |
| CONNECTION=          | SHAREDREAD                                                                           | •                |
| CONNECTION_GROUP=    | none                                                                                 | •                |
| CONOPTS=             | none                                                                                 | •                |
| DBCLIENT_MAX_BYTES=  | matches the maximum number of bytes per single character of the SAS session encoding | •                |
| DBCOMMIT=            | 1000 (when inserting rows), 0 (when updating rows)                                   |                  |
| DBCONINIT=           | none                                                                                 | •                |
| DBCONTERM=           | none                                                                                 | •                |
| DBCREATE_TABLE_OPTS= | none                                                                                 |                  |
| DBGEN_NAME=          | DBMS                                                                                 | •                |
| DBINDEX=             | NO                                                                                   |                  |
| DBLIBINIT=           | none                                                                                 |                  |
| DBLIBTERM=           | none                                                                                 |                  |
| DBMAX_TEXT=          | 1024                                                                                 | •                |
| DBMSTEMP=            | NO                                                                                   |                  |
| DBNULLKEYS=          | YES                                                                                  |                  |
| DBPROMPT=            | NO                                                                                   |                  |
| DBSERVER_MAX_BYTES=  | none                                                                                 | •                |
| DEFER=               | NO                                                                                   | •                |
| DELETE_MULT_ROWS=    | NO                                                                                   | •                |
| DIRECT_EXE=          | none                                                                                 |                  |
| DIRECT_SQL=          | YES                                                                                  |                  |

| Option                    | Default Value                                                | Valid in CONNECT |
|---------------------------|--------------------------------------------------------------|------------------|
| IGNORE_READ_ONLY_COLUMNS= | NO                                                           |                  |
| INSERTBUFF=               | 1                                                            |                  |
| LOGIN_TIMEOUT=            | 0                                                            | •                |
| MULTI_DATASRC_OPT=        | NONE                                                         |                  |
| PRESERVE_COL_NAMES=       | YES                                                          |                  |
| PRESERVE_TAB_NAMES=       | YES                                                          |                  |
| QUERY_TIMEOUT=            | 0                                                            | •                |
| QUOTE_CHAR=               | none                                                         | •                |
| READ_ISOLATION_LEVEL=     | RC (see “Locking in the Yellowbrick Interface” on page 1412) | •                |
| READ_LOCK_TYPE=           | ROW                                                          | •                |
| READBUFF=                 | 0                                                            | •                |
| REREAD_EXPOSURE=          | NO                                                           | •                |
| SCHEMA=                   | none                                                         |                  |
| SPOOL=                    | YES                                                          |                  |
| SQL_FUNCTIONS=            | none                                                         |                  |
| SQL_FUNCTIONS_COPY=       | none                                                         |                  |
| SSLMODE=                  | prefer                                                       |                  |
| STRINGDATES=              | NO                                                           | •                |
| TRACE=                    | NO                                                           | •                |
| TRACEFILE=                | none                                                         | •                |
| UPDATE_ISOLATION_LEVEL=   | RC (see “Locking in the Yellowbrick Interface” on page 1412) | •                |
| UPDATE_LOCK_TYPE=         | ROW                                                          | •                |

| Option              | Default Value | Valid in CONNECT |
|---------------------|---------------|------------------|
| UPDATE_MULT_ROWS=   | NO            |                  |
| UTILCONN_TRANSIENT= | NO            |                  |

---

## LIBNAME Statement Examples for Yellowbrick

In this example, SERVER=, DATABASE=, USER=, and PASSWORD= are connection options. No DSN style is specified. This is the default method, which is recommended.

```
libname A1 ybrick server=mysrv1 port=5432
      user=myusr1 password='mypwd1' database=mydb1;
```

This example requires that you specify a DSN style.

```
libname B1 ybrick dsn=ptgtest user=myusr1 password=mypwd1;
```

---

## Data Set Options for Yellowbrick

All SAS/ACCESS data set options in this table are supported for Yellowbrick. Default values are provided where applicable. For more information, see [Data Set Options for Relational Databases on page 370](#).

**Table 41.2** SAS/ACCESS Data Set Options for Yellowbrick

| Option              | Default Value                   |
|---------------------|---------------------------------|
| BL_DATAFILE=        | none                            |
| BL_DEFAULT_DIR=     | <i>temporary-file-directory</i> |
| BL_DELETE_DATAFILE= | YES                             |
| BL_DELIMITER=       | (pipe)                          |
| BL_ESCAPE=          | \                               |
| BL_FORMAT=          | TEXT                            |
| BL_LOGFILE=         | none                            |

| Option                    | Default Value                                        |
|---------------------------|------------------------------------------------------|
| BL_NULL=                  | '\N' [TEXT mode], unquoted empty value<br>[CSV mode] |
| BL_QUOTE=                 | " (double quotation mark)                            |
| BL_YB_PATH=               | none                                                 |
| BULKLOAD=                 | NO                                                   |
| BULKUNLOAD=               | NO                                                   |
| DBCOMMIT=                 | LIBNAME option value                                 |
| DBCONDITION=              | none                                                 |
| DBCREATE_TABLE_OPTS=      | LIBNAME option value                                 |
| DBFORCE=                  | NO                                                   |
| DBGEN_NAME=               | DBMS                                                 |
| DBINDEX=                  | LIBNAME option value                                 |
| DBKEY=                    | none                                                 |
| DBLABEL=                  | NO                                                   |
| DBLARGETABLE=             | none                                                 |
| DBMAX_TEXT=               | 1024                                                 |
| DBNULL=                   | YES                                                  |
| DBNULLKEYS=               | LIBNAME option value                                 |
| DBPROMPT=                 | LIBNAME option value                                 |
| DBSASLABEL=               | COMPAT                                               |
| DBSASTYPE=                | see “Data Types for Yellowbrick” on page<br>1414     |
| DBTYPE=                   | see “Data Types for Yellowbrick” on page<br>1414     |
| ERRLIMIT=                 | 1                                                    |
| IGNORE_READ_ONLY_COLUMNS= | NO                                                   |

| Option                  | Default Value        |
|-------------------------|----------------------|
| INSERTBUFF=             | LIBNAME option value |
| NULLCHAR=               | SAS                  |
| NULLCHARVAL=            | a blank character    |
| PG_IDENTITY_COLS=       | none                 |
| PRESERVE_COL_NAMES=     | LIBNAME option value |
| QUERY_TIMEOUT=          | LIBNAME option value |
| READBUFF=               | LIBNAME option value |
| READ_LOCK_TYPE=         | LIBNAME option value |
| SASDATEFMT=             | none                 |
| SCHEMA=                 | LIBNAME option value |
| UPDATE_ISOLATION_LEVEL= | LIBNAME option value |
| UPDATE_LOCK_TYPE=       | LIBNAME option value |
| YB_JAVA_HOME=           | none                 |

---

## Known Limitation When Working with Yellowbrick

SAS/ACCESS Interface to Yellowbrick does not support positional inserts, updates, or deletions. Therefore, the MODIFY statement in a DATA step is not supported for Yellowbrick.

---

# SQL Pass-Through Facility Specifics for Yellowbrick

---

## Key Information

For general information about this feature, see “[SQL Pass-Through Facility](#)” on page [690](#).

Here are the SQL pass-through facility specifics for the Yellowbrick interface.

- The *dbms-name* is `YBRICK`.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Yellowbrick. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default `YBRICK` alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME [connection options](#).
- To execute UPDATE and DELETE statements in PROC SQL, the DBIDIRECTEXEC system option must be enabled. (This is the default value for this option.)

---

## CONNECT Statement Example for Yellowbrick

This example connects to Yellowbrick and then disconnects from it.

```
proc sql noerrorstop;
    connect to ybrick as x1(server=mysrv1 port=5432
                           user=mysur1 password='mypwd1' database=mydb1);
    disconnect from x1;
    quit;
```

---

## Temporary Table Support for Yellowbrick

SAS/ACCESS Interface to Yellowbrick supports temporary tables. For more information, see “[Temporary Table Support for SAS/ACCESS](#)” on page [51](#).

# Passing SAS Functions to Yellowbrick

SAS/ACCESS Interface to Yellowbrick passes the following SAS functions to Yellowbrick for processing. Where the Yellowbrick function name differs from the SAS function name, the Yellowbrick name appears in parentheses. For more information, see “[Passing Functions to the DBMS Using PROC SQL](#)” on page 58.

|                            |                   |
|----------------------------|-------------------|
| ABS                        | LOG2 (LOG)        |
| ARCOS (ACOS)               | LOWCASE (LOWER)   |
| ARSIN (ASIN)               | MINUTE            |
| ATAN                       | MOD (see note)    |
| ATAN2                      | MONTH             |
| BYTE                       | QTR               |
| CEIL                       | REPEAT            |
| COALESCE                   | SECOND            |
| COS                        | SIGN              |
| COUNT                      | SIN               |
| DAY                        | SQRT              |
| DAYOFWEEK                  | STD (STDDEV_SAMP) |
| EXP                        | STRIP (BTRIM)     |
| FLOOR                      | SUBSTR            |
| HOUR                       | TAN               |
| INDEX (POSITION, STRPOS)   | TRANWRD (REPLACE) |
| LENGTH                     | TRIMN (RTRIM)     |
| LENGTHC (CHARACTER_LENGTH) | UPCASE (UPPER)    |
| LENGTHN (LENGTH)           | VAR (VAR_SAMP)    |
| LOG (LN)                   | YEAR              |
| LOG10 (LOG)                |                   |

---

**Note:** SAS does not modify non-integer arguments to the MOD function. If your DBMS does truncate non-integer arguments to MOD, then DBMS results for this function might vary from SAS results. For more information, see “[Functions Where Results Might Vary: MOD Function](#)” on page 59.

---

**SQL\_FUNCTIONS=ALL** allows for SAS functions that have slightly different behavior from corresponding database functions that are passed down to the database. Only when **SQL\_FUNCTIONS=ALL** can the SAS/ACCESS engine also pass these SAS SQL functions to Yellowbrick. Because of incompatibility in date and time functions between Yellowbrick and SAS, Yellowbrick might not process them correctly. Check your results to determine whether these functions are working as expected.

|                     |                      |
|---------------------|----------------------|
| COMPRESS            | ROUND                |
| DATE (CURRENT_DATE) | TIMEPART             |
| DATEPART            | TODAY (CURRENT_DATE) |

DATETIME (NOW)

TRANSLATE

---

## Passing Joins to Yellowbrick

For a multiple libref join to pass to Yellowbrick, all of these components of the LIBNAME statements must match exactly:

- user ID (USER=)
- password (PASSWORD=)
- server (SERVER=)
- database (DATABASE=)
- port (PORT=)
- SQL functions (SQL\_FUNCTIONS=)

For more information about when and how SAS/ACCESS passes joins to the DBMS, see [“Passing Joins to the DBMS” on page 61](#).

---

## Bulk Loading and Unloading for Yellowbrick

---

### Overview of Bulk Loading and Unloading for Yellowbrick

Bulk loading is the fastest way to insert large numbers of rows into a Yellowbrick table. To use the bulk-load utility, set the **BULKLOAD= LIBNAME** option or data set option to YES. Similarly, bulk unloading is the fastest way to retrieve large numbers of rows from a Yellowbrick table. To use the bulk-unloading utility, set the **BULKUNLOAD= LIBNAME** option or data set option to YES. Bulk loading is implemented with the Yellowbrick `yload` utility, which moves data from SAS into the Yellowbrick database. Bulk unloading is implemented with the Yellowbrick `yunload`, which moves data from Yellowbrick into SAS.

---

## LIBNAME and Data Set Options with Bulk Loading and Unloading

This section lists the Yellowbrick options for bulk loading and bulk unloading. For more information about these options, see “[Overview](#)” on page 370.

Here are the data set options for bulk loading and bulk unloading:

- [BL\\_DATAFILE=](#)
- [BL\\_DEFAULT\\_DIR=](#)
- [BL\\_DELETE\\_DATAFILE=](#)
- [BL\\_DELIMITER=](#)
- [BL\\_ESCAPE=](#)
- [BL\\_FORMAT=](#)
- [BL\\_LOGFILE=](#)
- [BL\\_NULL=](#)
- [BL\\_QUOTE=](#)
- [BL\\_YB\\_PATH=](#)
- [BULKLOAD=](#)
- [BULKUNLOAD=](#)
- [YB\\_JAVA\\_HOME=](#)

---

## Bulk-Load Example for Yellowbrick

This first example shows how you can use a SAS data set, SASFLT.FLT98, to create and load a large Yellowbrick table, FLIGHTS98:

```
libname sasflt 'SAS-library';
libname net_air ybrick user=myusr1 pwd=mypwd1
    server=air2 database=flights;

proc sql;
create table net_air.flights98 (bulkload=YES bl_yb_path='path-to-
yblast')
    as select * from sasflt.flt98;
quit;
```

# Locking in the Yellowbrick Interface

The following LIBNAME and data set options let you control how the Yellowbrick interface handles locking. For general information about an option, see “[LIBNAME Options for Relational Databases](#)” on page 134.

`READ_LOCK_TYPE= ROW`

`UPDATE_LOCK_TYPE= ROW`

`READ_ISOLATION_LEVEL= S | RC`

The Yellowbrick ODBC driver manager supports the S and RC isolation levels that are specified in this table.

*Table 41.3 Isolation Levels for Yellowbrick*

| Isolation Level       | Definition                                                                                                                                                                                                                       |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| S (serializable)      | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads.                                                                                                                                                               |
| RR (repeatable read)  | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.                                                                                                                                                     |
| RC (read committed)   | Does not allow dirty Reads or nonrepeatable Reads; does allow phantom Reads.                                                                                                                                                     |
| RU (read uncommitted) | Allows dirty Reads, nonrepeatable Reads, and phantom Reads,                                                                                                                                                                      |
| V (versioning)        | Does not allow dirty Reads, nonrepeatable Reads, or phantom Reads. These transactions are serializable, but higher concurrency is possible than with the serializable isolation level. Typically, a nonlocking protocol is used. |

Here are how the terms in the table are specified.

#### *Dirty read*

A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, this type of transaction sees changes that are made by concurrent transactions even before they are committed.

For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

#### *Phantom read*

When a transaction exhibits this phenomenon, a set of rows that the transaction reads once might be a different set of rows if the transaction attempts to read them again.

For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it sees a row that did not previously exist, a phantom.

UPDATE\_ISOLATION\_LEVEL= S | RC

The Yellowbrick ODBC driver manager supports the S and RC isolation levels that are specified in the preceding table.

---

## Naming Conventions for Yellowbrick

For general information, see [Chapter 3, “SAS Names and Support for DBMS Names,” on page 21](#).

Most SAS names can be up to 32 characters long. The Yellowbrick interface supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less. SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The PRESERVE\_TAB\_NAMES= and PRESERVE\_COL\_NAMES= options determine how this interface handles case sensitivity, spaces, and special characters. (For information about these options, see [LIBNAME Statement for Relational Databases on page 127](#).) Yellowbrick is not case sensitive, and all names default to lowercase.

Yellowbrick objects include tables, views, and columns. They follow these naming conventions.

- A name can contain up to 128 characters.
- The first character in a name can be a letter, @, \_, or #.
- A name cannot be a Yellowbrick reserved word, such as WHERE or VIEW.
- A name cannot be the same as another Yellowbrick object that has the same type.
- A name must be unique within each type of each object.

---

# Data Types for Yellowbrick

---

## Overview of Data Types

Every column in a table has a name and a data type. The data type tells Yellowbrick how much physical storage to set aside for the column and the form in which the data is stored. This section includes information about Yellowbrick data types, null and default values, and data conversions.

For more information about Yellowbrick data types and to determine which data types are available for your version of Yellowbrick, see your Yellowbrick documentation.

---

## Yellowbrick Null Values

Yellowbrick has a special value called NULL. A Yellowbrick NULL value signifies an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Yellowbrick NULL value, it interprets it as a SAS missing value.

You can define a column in a Yellowbrick table so that it requires data. To do this in SQL, you specify a column as NOT NULL. This tells SQL to allow a row to be added to a table only if a value exists for the field. For example, when NOT NULL is assigned to the CUSTOMER field in the SASDEMO.CUSTOMER table, a row cannot be added unless there is a value for CUSTOMER. When creating a table with SAS/ACCESS, you can use the DBNULL= data set option to indicate whether NULL is a valid value for specified columns.

You can also specify Yellowbrick columns as NOT NULL DEFAULT. For more information about using the NOT NULL DEFAULT value, see your Yellowbrick documentation.

When you know whether a Yellowbrick column enables NULLs or the host system provides a default value for a column that is specified as NOT NULL WITH DEFAULT, you can write selection criteria and enter values to update a table. Unless a column is specified as NOT NULL or NOT NULL DEFAULT, it allows NULL values.

For more information about how SAS handles NULL values, see “[Potential Result Set Differences When Processing Null Data](#)” on page 43.

To control how SAS missing character values are handled, use the **NULLCHAR=** and **NULLCHARVAL=** data set options.

# Working with Long Character Values in Yellowbrick

The default open-source driver automatically assigns a length of 255 characters to varying-length character variables, such as the VARCHAR(*n*) data type. Therefore, if you know that character data in a data set is longer than 255 characters, you should specify the MaxVarcharSize attribute for the CONOPTS= LIBNAME option when you connect to your Yellowbrick library. Specify a value for MaxVarcharSize that is as large as the longest column value to ensure that you retrieve all of the available data.

The following LIBNAME statement, sample tables, and SQL query would create a new data set that does not truncate the character data from the Testb data set.

```

libname x ybrick preserve_col_names=no
preserve_tab_names=no database=mydb1 dbmax_text=32767
server='myserver' port=5432 user=ybadmin password='ybpwd'
conopts='MaxVarcharSize=300';

/* Create example tables Testa and Testb */
proc sql;
create table testa(idnum int, myflag int, col1 varchar(20))
create table testb(idnum int, col1 varchar(300))

insert into testa values (1, 1, 'from a 1')
insert into testa values (2, 0, 'from a 2')

insert into testb values (1, 'from b 1*from b 1*from b 1*from b 1*from
b 1*
      from b 1*from b 1*from b 1*from b 1*from b 1*from b 1*from
b 1*
      from b 1*from b 1*from b 1*from b 1*from b 1*from b 1*from
b 1*
      from b 1*from b 1*from b 1*from b 1*from b 1*from b 1*from
b 1*
      from b 1*274')
insert into testb values (2, 'from b 2*from b 2*from b 2*from b 2*from
b 2*
      from b 2*from b 2*from b 2*from b 2*from b 2*from b 2*from
b 2*
      from b 2*from b 2*from b 2*from b 2*from b 2*from b 2*from
b 2*
      from b 2*from b 2*from b 2*from b 2*from b 2*from b 2*from
b 2*
      from b 2*274')
quit;

/* Verify that data is not truncated */
proc sql;
create table work.SASTEST as
select case when myflag = 1 then a.col1 else b.col1 end as
      col1 from x.testa a, x.testb b
      where a.idnum = b.idnum;

```

```

/* Select the end of col1 from position 200 to the end to show that
all of */
/* the data is
retrieved
select substr(col1, 200) from work.SASTEST;
quit;

```

The output from the final SELECT statement appears as shown here. This statement selects a substring starting at column position 200 from the Col1 value.

```
rom b 2*from b 2*274
```

You can see that the end of the value does include the ‘\*274’ from the end of the VARCHAR value.

## LIBNAME Statement Data Conversions

This table shows the default formats that SAS/ACCESS Interface to Yellowbrick assigns to SAS variables when using the [LIBNAME statement](#) to read from a Yellowbrick table. These default formats are based on Yellowbrick column attributes. SAS/ACCESS does not support Yellowbrick data types that do not appear in this table.

**Table 41.4 LIBNAME Statement: Default SAS Formats for Yellowbrick Data Types**

| Yellowbrick Data Type            | ODBC Data Type           | Default SAS Format                                                                      |
|----------------------------------|--------------------------|-----------------------------------------------------------------------------------------|
| CHAR( <i>n</i> ) <sup>1</sup>    | SQL_CHAR                 | \$ <i>w</i> .                                                                           |
| VARCHAR( <i>n</i> ) <sup>1</sup> | SQL_LONGCHAR( <i>n</i> ) |                                                                                         |
| DECIMAL                          | SQL_DECIMAL              | <i>w</i> or <i>w.d</i> or none if you do not specify <i>w</i> and <i>d</i> <sup>2</sup> |
| NUMERIC                          | SQL_NUMERIC              |                                                                                         |
| INTEGER                          | SQL_INTEGER              | 11.                                                                                     |
| SMALLINT                         | SQL_SMALLINT             | 6.                                                                                      |
|                                  | SQL_TINYINT              | 4.                                                                                      |
| REAL                             | SQL_REAL                 | none                                                                                    |
| FLOAT                            | SQL_FLOAT                |                                                                                         |
| DOUBLE PRECISION                 | SQL_DOUBLE               |                                                                                         |
| BIGINT                           | SQL_BIGINT               | 20.                                                                                     |
| UUID                             | SQL_GUID                 | \$ <i>w</i> .                                                                           |

| Yellowbrick Data Type | ODBC Data Type | Default SAS Format |
|-----------------------|----------------|--------------------|
| DATE                  | SQL_TYPE_DATE  | DATE9.             |

- 1  $n$  in Yellowbrick character data types is equivalent to  $w$  in SAS formats.  
 2  $m$  and  $n$  in Yellowbrick numeric data types are equivalent to  $w$  and  $d$  in SAS formats.

The following table shows the default data types that SAS/ACCESS Interface to Yellowbrick uses when creating tables. The Yellowbrick engine lets you specify nondefault data types by using the DBTYPE= data set option.

**Table 41.5 LIBNAME Statement: Default SAS Formats for Yellowbrick Data Types When Creating Tables**

| Yellowbrick Data Type         | ODBC Data Type                                              | Default SAS Format |
|-------------------------------|-------------------------------------------------------------|--------------------|
| NUMERIC( $m,n$ ) <sup>1</sup> | SQL_DOUBLE or SQL_NUMERIC using $m,n$ if the DBMS allows it | $w.d$              |
| VARCHAR( $n$ ) <sup>2</sup>   | SQL_VARCHAR using $n$                                       | \$ $w$ .           |
| TIMESTAMP <sup>3</sup>        | SQL_TIMESTAMP                                               | DATETIME formats   |
| DATE                          | SQL_DATE                                                    | DATE formats       |
| TIME                          | SQL_TIME                                                    | TIME formats       |

- 1  $m$  and  $n$  in Yellowbrick numeric data types are equivalent to  $w$  and  $d$  in SAS formats.  
 2  $n$  in Yellowbrick character data types is equivalent to  $w$  in SAS formats.  
 3 Although the Yellowbrick engine supports TIMESTAMP, it has no TIMESTAMP WITH TIMEZONE data type that maps to the corresponding Yellowbrick data type.

## Items of Note for Yellowbrick Data

When you work with data from Yellowbrick, here are some items to keep in mind:

- Table indexes are not supported.
- Foreign keys are not supported.
- Primary and unique key constraints can be specified, but they are not enforced.
- Yellowbrick data storage is based on bytes rather than characters. For this reason, use the DBSERVER\_MAX\_BYTES= LIBNAME option and the DBCLIENT\_MAX\_BYTES= LIBNAME option to control data storage sizes.



**PART 4****Appendices**

|                                              |      |
|----------------------------------------------|------|
| Appendix 1<br><i>ACCESS Procedure</i> .....  | 1421 |
| Appendix 2<br><i>CV2VIEW Procedure</i> ..... | 1443 |
| Appendix 3<br><i>DBLOAD Procedure</i> .....  | 1455 |



# Chapter 42

## ACCESS Procedure

---

|                                                       |             |
|-------------------------------------------------------|-------------|
| <b>Overview: ACCESS Procedure . . . . .</b>           | <b>1421</b> |
| Accessing DBMS Data . . . . .                         | 1422        |
| About ACCESS Procedure Statements . . . . .           | 1422        |
| <b>Concepts: ACCESS Procedure . . . . .</b>           | <b>1424</b> |
| DBMS Specifics: ACCESS Procedure . . . . .            | 1424        |
| <b>Syntax: ACCESS Procedure . . . . .</b>             | <b>1424</b> |
| PROC ACCESS Statement . . . . .                       | 1425        |
| Database Connection Statement . . . . .               | 1426        |
| ASSIGN Statement . . . . .                            | 1426        |
| CREATE Statement . . . . .                            | 1427        |
| DROP Statement . . . . .                              | 1429        |
| FORMAT Statement . . . . .                            | 1429        |
| LIST Statement . . . . .                              | 1430        |
| QUIT Statement . . . . .                              | 1431        |
| RENAME Statement . . . . .                            | 1432        |
| RESET Statement . . . . .                             | 1433        |
| SELECT Statement . . . . .                            | 1434        |
| SUBSET Statement . . . . .                            | 1435        |
| TABLE Statement . . . . .                             | 1436        |
| UNIQUE Statement . . . . .                            | 1436        |
| UPDATE Statement . . . . .                            | 1437        |
| <b>Usage: ACCESS Procedure . . . . .</b>              | <b>1439</b> |
| Using Descriptors with the ACCESS Procedure . . . . . | 1439        |
| <b>Examples: ACCESS Procedure . . . . .</b>           | <b>1441</b> |
| Example 1: Update an Access Descriptor . . . . .      | 1441        |
| Example 2: Create a View Descriptor . . . . .         | 1441        |

# Overview: ACCESS Procedure

## Accessing DBMS Data

SAS still supports this legacy procedure. However, to access your relational DBMS data more directly, it is recommended that you use the [SAS/ACCESS LIBNAME statement on page 127](#) or the [SQL pass-through facility on page 690](#). To determine whether this feature is available in your environment for your interface, see [SAS/ACCESS Features by Host on page 96](#).

With the DBLOAD procedure and an interface view engine, the ACCESS procedure creates an interface between SAS and data in vendor databases. You can use this procedure to create and update descriptors.

## About ACCESS Procedure Statements

This procedure has several types of statements:

- *database connection statements*: for connecting to your DBMS (see DBMS-specific information in this document for your SAS/ACCESS interface)
- *creating and updating statements*: [CREATE](#) and [UPDATE](#)
- *table and editing statements*: [ASSIGN](#), [DROP](#), [FORMAT](#), [LIST](#), [QUIT](#), [RENAME](#), [RESET](#), [SELECT](#), [SUBSET](#), [TABLE](#), and [UNIQUE](#)

This table summarizes PROC ACCESS options and statements that are required to accomplish common tasks.

**Table A5.1** Statement Sequence for Accomplishing Tasks with the ACCESS Procedure

| Task                                              | Statements and Options to Use                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Create an access descriptor                       | <b>PROC ACCESS</b> <i>statement-options</i> ;<br><b>CREATE</b> <i>libref.member-name.ACCESS</i> ;<br><i>database-connection-statements</i> ;<br><i>editing-statements</i> ;<br><b>RUN</b> ;;                                   |
| Create an access descriptor and a view descriptor | <b>PROC ACCESS</b> <i>statement-options</i> ;<br><b>CREATE</b> <i>libref.member-name.ACCESS</i> ;<br><i>database-connection-statements</i> ;<br><i>editing-statements</i> ;<br><b>CREATE</b> <i>libref.member-name.VIEW</i> ;; |

| Task                                                        | Statements and Options to Use                                                                                                                                                                                                                   |
|-------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                             | <pre><b>SELECT</b> column-list; editing-statements; <b>RUN</b>;</pre>                                                                                                                                                                           |
| Create a view descriptor from an existing access descriptor | <pre><b>PROC ACCESS</b> statement-options, including ACCDESC=libref.access-descriptor; <b>CREATE</b> libref.member-name.VIEW; <b>SELECT</b> column-list; editing-statements; <b>RUN</b>;</pre>                                                  |
| Update an access descriptor                                 | <pre><b>PROC ACCESS</b> statement-options; <b>UPDATE</b> libref.member-name.ACCESS; database-connection-statements; editing-statements; <b>RUN</b>;</pre>                                                                                       |
| Update an access descriptor and a view descriptor           | <pre><b>PROC ACCESS</b> statement-options; <b>UPDATE</b> libref.member-name.ACCESS; database-connection-statements; editing-statements; <b>UPDATE</b> libref.member-name.VIEW; editing-statements; <b>RUN</b>;</pre>                            |
| Update an access descriptor and create a view descriptor    | <pre><b>PROC ACCESS</b> statement-options; <b>UPDATE</b> libref.member-name.ACCESS; database-connection-statements; editing-statements; <b>CREATE</b> libref.member-name.VIEW; <b>SELECT</b> column-list; editing-statements; <b>RUN</b>;</pre> |
| Update a view descriptor from an existing access descriptor | <pre><b>PROC ACCESS</b> statement-options, including ACCDESC=libref.access-descriptor; <b>UPDATE</b> libref.member-name.VIEW; editing-statements; <b>RUN</b>;</pre>                                                                             |
| Create a SAS data set from a view descriptor                | <pre><b>PROC ACCESS</b> statement-options, including DBMS=dbms-name; VIEWDESC=libref.member; OUT=libref.member; <b>RUN</b>;</pre>                                                                                                               |

---

# Concepts: ACCESS Procedure

---

## DBMS Specifics: ACCESS Procedure

These SAS/ACCESS interfaces support the ACCESS procedure:

- DB2 z/OS
- Oracle
- SAP ASE

---

## Syntax: ACCESS Procedure

See: [DB2 z/OS, Oracle, SAP ASE](#)

```
PROC ACCESS <options>;
  database-connection-statements;
CREATE libref.member-name.libref.member-name.ACCESS | VIEW password-
  option ;
UPDATE libref.member-name.VIEW <password-option>;
TABLE=<'>table-name<'>;
ASSIGN <=> YES | NO | Y | N;
DROP <'>column-identifier-1 <'> <...<'>column-identifier-n<'>>;
FORMAT <'>column-identifier-1 <'> <=>SAS-format-name-1
  <...<'>column-identifier-n <'> <=> SAS-format-name-n>;
LIST <ALL | VIEW | <'>column-identifier<'>>;
QUIT;
RENAME <'>column-identifier-1 <'> <=> SAS-variable-name-1 <...<'>column-
  identifier-n <'>
  <=> SAS-variable-name-n>;
RESET ALL | <'>column-identifier-1 <'> <...<'>column-identifier-n <'>>;
SELECT ALL | <'>column-identifier-1 <'> <...<'>column-identifier-n <'>>;
SUBSET selection-criteria;
UNIQUE <=> YES | NO | Y | N;
RUN;
```

| Statement           | Task                                                             |
|---------------------|------------------------------------------------------------------|
| PROC ACCESS         | Access relational DBMS data                                      |
| Database Connection | Provide DBMS-specific connection information                     |
| ASSIGN              | Indicate whether SAS variable names and formats are generated    |
| CREATE              | Create a SAS/ACCESS descriptor file                              |
| DROP                | Drop a column so that it cannot be selected in a view descriptor |
| FORMAT              | Change a SAS format for a DBMS column                            |
| LIST                | List columns in the descriptor and give information about them   |
| QUIT                | Terminate the procedure                                          |
| RENAME              | Modify the SAS variable name                                     |
| RESET               | Reset DBMS columns to their default values                       |
| SELECT              | Select DBMS columns for the view descriptor                      |
| SUBSET              | Add or modify selection criteria for a view descriptor           |
| TABLE               | Identify the DBMS table on which the access descriptor is based  |
| UNIQUE              | Generate SAS variable names based on DBMS column names           |
| UPDATE              | Update a SAS/ACCESS descriptor file                              |

---

## PROC ACCESS Statement

Accesses relational DBMS data

---

## Syntax

**PROC ACCESS <options>;**

## Optional Arguments

**ACCDESC=libref.access-descriptor**

specifies an access descriptor. ACCDESC= is used with the DBMS= option to create or update a view descriptor that is based on the specified access descriptor. You can use a SAS data set option on the ACCDESC= option to specify any passwords that have been assigned to the access descriptor.

**Restriction** The ODBC interface does not support this option.

**DBMS=database-management-system**

specifies which database management system you want to use. This DBMS-specific option is required. For details, see the DBMS-specific information for your SAS/ACCESS interface.

**OUT=libref.member-name**

specifies the SAS data set in which DBMS data is stored.

**VIEWDESC=libref.view-descriptor**

specifies a view descriptor through which you extract the DBMS data.

## Database Connection Statement

Provides DBMS-specific connection information

See: *Database connection statements* are used to connect to your DBMS. For details, see the DBMS-specific information for your SAS/ACCESS interface.

## Syntax

*database-connection-statements;*

## ASSIGN Statement

Indicates whether SAS variable names and formats are generated

Default: NO

Interactions: FORMAT, RENAME, RESET, UNIQUE  
applies to access descriptor

## Syntax

**ASSIGN <=> YES | NO;**

## Required Arguments

### YES

generates unique SAS variable names from the first eight characters of the DBMS column names. If you specify YES, you cannot specify the RENAME, FORMAT, RESET, or UNIQUE statements when you create view descriptors that are based on the access descriptor.

Alias Y

### NO

lets you modify SAS variable names and formats when you create an access descriptor and when you create view descriptors that are based on this access descriptor.

Alias N

## Details

The ASSIGN statement indicates how SAS variable names and formats are assigned:

- SAS automatically generates SAS variable names.
- You can change SAS variable names and formats in the view descriptors that are created from the access descriptor.

Each time the SAS/ACCESS interface encounters a CREATE statement to create an access descriptor, the ASSIGN statement is reset to the default NO value.

When you create an access descriptor, use the RENAME statement to change SAS variable names and the FORMAT statement to change SAS formats.

When you specify YES, SAS generates names according to these rules:

- You can change the SAS variable names only in the access descriptor.
- SAS variable names that are saved in an access descriptor are *always* used when view descriptors are created from the access descriptor. You cannot change them in the view descriptors.
- The ACCESS procedure allows names only up to eight characters.

## CREATE Statement

Creates a SAS/ACCESS descriptor file

Requirement: This statement is required.

Interaction: applies to access descriptor or view descriptor

See: [Statement Sequence for Accomplishing Tasks on page 1422](#)

## Syntax

**CREATE** *libref.member-name.ACCESS* | **VIEW** <*password-option*>;

### Required Arguments

***libref.member-name***

identifies the libref of the SAS library where you want to store the descriptor and identifies the descriptor name.

**ACCESS**

specifies an access descriptor.

**VIEW**

specifies a view descriptor.

### Optional Argument

***password-option***

specifies a password.

---

## Details

This statement names the access descriptor or view descriptor that you are creating. Use a three-level name:

- The first level identifies the libref of the SAS library where you want to store the descriptor.
- The second level is the descriptor name.
- The third level specifies the type of SAS file (specify **ACCESS** for an access descriptor or **VIEW** for a view descriptor).

---

## Example

Within the same PROC ACCESS step, this example creates an AdLib.Employ access descriptor on the Employees Oracle table and a Vlib.Emp1204 view descriptor based on AdLib.Employ.

```
proc access dbms=oracle;

/* create access descriptor */
create adlib.employ.access;
database='qa:[dubois]textile';
table=employees;
assign=no;
list all;

/* create view descriptor */
create vlib.emp1204.view;
```

```

select empid lastname hiredate salary dept
      gender birthdate;
      format empid 6.
                  salary dollar12.2
                  jobcode 5.
                  hiredate datetime9.
                  birthdate datetime9. ;
      subset where jobcode=1204;
run;

```

## DROP Statement

Drops a column so that it cannot be selected in a view descriptor

Interactions:     **RESET, SELECT**  
applies to access or view descriptors

## Syntax

**DROP <'>column-identifier-1<'> <...<'>column-identifier-n<'>>;**

### Required Argument

#### *column-identifier*

specifies the column name or the positional equivalent from the LIST statement.  
This is the number that represents the column's place in the access descriptor.

For example, to drop the third and fifth columns, submit this statement: `drop 3  
5 ;`

## Details

The DROP statement drops one or more specified columns from a descriptor. You can drop a column when you create or update an access descriptor. You can also drop a column when you update a view descriptor. If you drop a column when you create an access descriptor, you cannot select that column when you create a view descriptor that is based on the access descriptor. This statement does not affect the underlying DBMS table.

To display a previously dropped column, specify that column name in the RESET statement. However, doing this also resets all column attributes, such as the SAS variable name and format, to their default values.

## FORMAT Statement

Changes a SAS format for a DBMS column

Interactions: ASSIGN, DROP, RESET  
applies to access or view descriptors

## Syntax

```
FORMAT <'>column-identifier-1<'>
<=>SAS-format-name-1 <...<'>column-identifier-n<'> <=> SAS-format-name-n>;
```

## Required Arguments

### **column-identifier**

specifies the column name or the positional equivalent from the LIST statement. This is the number that represents the column's place in the access descriptor. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotation marks.

### **SAS-format-name**

specifies the SAS format to be used.

## Details

The FORMAT statement changes SAS variable formats from their default formats. The default SAS variable format is based on the data type of the DBMS column. For details about default formats that SAS assigns to your DBMS data types, see the DBMS-specific information for your SAS/ACCESS interface.

You can use the FORMAT statement with a view descriptor only if the ASSIGN statement that was used when creating the access descriptor was specified with the NO value. When you use the FORMAT statement with access descriptors, the FORMAT statement also reselects columns that were previously dropped with the DROP statement.

For example, submit this statement to associate the DATE9. format with the BIRTHDATE column and with the second column in the access descriptor:

```
format 2=date9. birthdate=date9.;
```

The equal sign (=) is optional. For example, you can use the FORMAT statement to specify new SAS variable formats for four DBMS table columns:

```
format productid 4.
      weight     e16.9
      fibersize e20.13
      width      e16.9;
```

## LIST Statement

Lists columns in the descriptor and gives information about them

Default: ALL

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Restrictions: | You can use this statement only when you create an access descriptor or a view descriptor.<br><br>Changes that were made since the last CREATE, UPDATE, or RUN statement are not saved. Changes are saved only when a new CREATE, UPDATE, or RUN statement is submitted.                                                                                                                                                                              |
| Interactions: | applies to access or view descriptors<br><br>To review the contents of an existing view descriptor, use the CONTENTS procedure.                                                                                                                                                                                                                                                                                                                       |
| Notes:        | LIST information is written to your SAS log.<br><br>When you use LIST for an access descriptor, *NON-DISPLAY* appears next to the column description for any column that has been dropped and *UNSUPPORTED* appears next to any column if your DBMS interface view engine does not support the data type. When you use LIST for a view descriptor, *SELECTED* appears next to the column description for columns that you have selected for the view. |
| Tip:          | Specify LIST last in your PROC ACCESS code to see the entire descriptor. If you create or update multiple descriptors, specify LIST before each CREATE or UPDATE statement to list information about all descriptors that you are creating or updating.                                                                                                                                                                                               |

---

## Syntax

**LIST <ALL | VIEW | <'>column-identifier<'>>;**

### Optional Arguments

#### **ALL**

lists all DBMS columns in the table, positional equivalents, SAS variable names, and SAS variable formats that are available for a descriptor.

#### **VIEW**

lists all DBMS columns that are selected for a view descriptor, their positional equivalents, their SAS names and formats, and any subsetting clauses.

#### **column-identifier**

lists information about a specified DBMS column, including its name, positional equivalent, SAS variable name and format, and whether it has been selected. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotation marks.

The *column-identifier* argument can be either the column name or the positional equivalent. This is the number that represents the column's place in the descriptor. For example, to list information about the fifth column in the descriptor, submit this statement:

```
list 5;
```

---

## QUIT Statement

Terminates the procedure without any further descriptor creation

Interaction: applies to access or view descriptors

## Syntax

**QUIT;**

---

## RENAME Statement

Specifies or modifies the SAS variable name that is associated with a DBMS column

Interactions:     **ASSIGN, RESET**  
                  applies to access or view descriptors

---

## Syntax

**RENAME <'>column-identifier-1<'> <=> SAS-variable-name-1 <...<'>  
column-identifier-n<'> <=> SAS-variable-name-n>;**

## Required Arguments

### **column-identifier**

specifies the DBMS column name or the positional equivalent from the LIST statement. This is the number that represents the column's place in the descriptor. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotation marks. The equal sign (=) is optional.

### **SAS-variable-name**

specifies a SAS variable name.

---

## Details

Two factors affect the use of the RENAME statement: whether you specify the ASSIGN statement when you are creating an access descriptor, and the type of descriptor that you are creating.

- If you omit the ASSIGN statement or specify it with a NO value, the renamed SAS variable names that you specify in the access descriptor are retained when an ACCESS procedure executes. For example, if you rename the CUSTOMER column to CUSTNUM when you create an access descriptor, the column is still named CUSTNUM when you select it in a view descriptor. The exception would be if you specified another RESET or RENAME statement.

When you create a view descriptor that is based on this access descriptor, you can specify the RESET statement or another RENAME statement to rename the variable. However, the new name applies only in that view. When you create other view descriptors, the SAS variable names are derived from the access descriptor.

- If you specify the YES value in the ASSIGN statement, you can use the RENAME statement to change SAS variable names only while creating an access descriptor. As described earlier, SAS variable names and formats that are saved in an access descriptor are always used when creating view descriptors that are based on the access descriptor.

For example, to rename the SAS variable names that are associated with the seventh column and the nine-character FIRSTNAME column in a descriptor, submit this statement:

```
rename
7 birthday 'firstname'=fname;
```

When you are creating a view descriptor, the RENAME statement automatically selects the renamed column for the view. That is, if you rename the SAS variable associated with a DBMS column, you do not have to issue a SELECT statement for that column.

## RESET Statement

Resets DBMS columns to their default values

Interactions:     ASSIGN, DROP, FORMAT, RENAME, SELECT  
                  applies to access or view descriptors

## Syntax

**RESET ALL | <'>column-identifier-1<'> <...<'>column-identifier-n<'>>;**

### Required Arguments

#### **ALL**

resets all columns in an access descriptor to their default names and formats and reselects any dropped columns. ALL deselects all columns in a view descriptor so that no columns are selected for the view.

#### ***column-identifier***

can be either the DBMS column name or the positional equivalent from the LIST statement. This is the number that represents the column's place in the access descriptor. If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotation marks. For example, to reset the SAS variable name and format associated with the third column, submit this statement: `reset 3;`

For access descriptors, the specified column is reset to its default name and format values. For view descriptors, the specified column is no longer selected for the view.

## Details

This statement has different effects on access and view descriptors.

For *access descriptors*, the RESET statement resets the specified column names to the default names that are generated by the ACCESS procedure. The RESET statement also changes the current SAS variable format to the default SAS format. Any previously dropped columns that are specified in the RESET statement become available.

When creating an access descriptor, if you omit the ASSIGN statement or set it to NO, the default SAS variable names are blanks. If you specify ASSIGN=YES, default names are the first eight characters of each DBMS column name.

For *view descriptors*, the RESET statement clears (deselects) any columns that were included in the SELECT statement. When you create a view descriptor that is based on an access descriptor that is created without an ASSIGN statement or with ASSIGN=NO, resetting and then reselecting (within the same procedure execution) a SAS variable changes the SAS variable names and formats to their default values. When you create a view descriptor that is based on an access descriptor created with ASSIGN=YES, the RESET statement does not have this effect.

---

## SELECT Statement

Selects DBMS columns for the view descriptor

Requirement: This statement is required.

Interactions: RESET  
applies to view descriptors

Note: The SELECT statement specifies which DBMS columns in an access descriptor to include in a view descriptor.

Tip: To clear your current selections when creating a view descriptor, use the RESET ALL statement.

---

## Syntax

**SELECT ALL | <'>column-identifier-1<'> <...<'>column-identifier-n<'>>;**

### Required Arguments

#### **ALL**

includes in the view descriptor all columns that were specified in the access descriptor and that were not dropped.

#### **column-identifier**

can be either the DBMS column name or the positional equivalent from the LIST statement. The positional equivalent is the number that represents where the column is located in the access descriptor on which the view is based. For example, to select the first three columns, submit this statement: `select 1 2 3;`

If the column name contains lowercase characters, special characters, or national characters, enclose the name in quotation marks.

---

## Details

SELECT statements are cumulative within a view creation. For example, columns 1, 5, and 6 are selected if you submit these SELECT statements:

```
select 1;
select 5 6;
```

---

## SUBSET Statement

Adds or modifies selection criteria for a view descriptor

- Restriction:** The SUBSET statement is case sensitive, unlike other ACCESS procedure statements. The SQL statement is sent to the DBMS exactly as you enter it. You must therefore use the correct case for any DBMS object names. For details, see the DBMS-specific information for your SAS/ACCESS interface.
- Interactions:** applies to view descriptors  
If you specify more than one SUBSET statement per view descriptor, the last SUBSET overwrites the earlier SUBSETS.
- Notes:** This statement is optional.  
SAS does not check the SUBSET statement for errors. The statement is verified only when the view descriptor is used in a SAS program.
- Tips:** You can use the SUBSET statement to specify selection criteria when you create a view descriptor.  
If you omit it, the view retrieves all data (rows) in the DBMS table.  
To delete the selection criteria, submit a SUBSET statement without any arguments.
- Example:** For a view descriptor that retrieves rows from a DBMS table, you could submit this statement:

```
subset where firstorder is not null;
```

---

## Syntax

**SUBSET** *selection-criteria*;

### Required Argument

***selection-criteria***

one or more DBMS-specific SQL expressions that are accepted by your DBMS, such as WHERE, ORDER BY, HAVING, and GROUP BY. Use DBMS column names, not SAS variable names, in your selection criteria.

## Details

If you have multiple selection criteria, enter them all in one SUBSET statement, as shown in this example:

```
subset where firstorder is not null
  and country = 'USA'
  order by country;
```

---

## TABLE Statement

Identifies the DBMS table on which the access descriptor is based

Interactions:     applies to access descriptors  
                   required with the CREATE statement  
                   optional with the UPDATE statement

---

## Syntax

**TABLE=<'>*table-name*<'>;**

### Required Argument

***table-name***

a valid DBMS table name. If it contains lowercase characters, special characters, or national characters, you must enclose it in quotation marks. For details, see the DBMS-specific information for your SAS/ACCESS interface.

---

## UNIQUE Statement

Generates SAS variable names based on DBMS column names

Interactions:     ASSIGN  
                   applies to view descriptors

Notes:           It is recommended that you use the UNIQUE statement and specify UNIQUE=YES. If you omit the UNIQUE statement or specify UNIQUE=NO and SAS encounters duplicate SAS variable names in a view descriptor, your job fails.  
                  The equal sign (=) is optional in the UNIQUE statement.

Tip:            If duplicate SAS variable names exist in the access descriptor on which you are creating a view descriptor, you can specify UNIQUE to resolve the duplication.

See:           [RENAME statement](#)

---

## Syntax

**UNIQUE <=> YES | NO;**

### Required Arguments

#### YES

causes the SAS/ACCESS interface to append numbers to any duplicate SAS variable names, making each variable name unique.

Alias Y

#### NO

causes the SAS/ACCESS interface to continue to allow duplicate SAS variable names to exist. You must resolve these duplicate names before saving (and thereby creating) the view descriptor.

Alias N

---

## Details

The UNIQUE statement specifies whether the SAS/ACCESS interface should generate unique SAS variable names for DBMS columns for which SAS variable names have not been entered.

The UNIQUE statement is affected by whether you specified the ASSIGN statement when you created the access descriptor on which the view is based:

- If you specified the ASSIGN=YES statement, you cannot specify UNIQUE when creating a view descriptor. YES causes SAS to generate unique names, so UNIQUE is not necessary.
- If you omitted the ASSIGN statement or specified ASSIGN=NO, you must resolve any duplicate SAS variable names in the view descriptor. You can use UNIQUE to generate unique names automatically, or you can use the RENAME statement to resolve duplicate names yourself.

---

## UPDATE Statement

Updates a SAS/ACCESS descriptor file

- Interactions: applies to access or view descriptors  
ASSIGN, RESET, SELECT, UNIQUE statements are not supported when you use the UPDATE statement.
- Note: Rules that apply to the CREATE statement also apply to the UPDATE statement. For example, the SUBSET statement is valid only for updating view descriptors.
- See: [Statement Sequence for Accomplishing Tasks on page 1422](#)

---

## Syntax

**UPDATE** *libref.member-name.YES | NO <password-option>;*

### Required Arguments

***libref.member-name***

identifies the libref of the SAS library where you want to store the descriptor and identifies the descriptor name.

**ACCESS**

specifies an access descriptor.

**VIEW**

specifies a view descriptor.

### Optional Argument

***password-option***

specifies a password.

---

## Details

The UPDATE statement identifies an existing access descriptor or view descriptor that you want to update. UPDATE is normally used to update database connection information, such as user IDs and passwords. If your descriptor requires many changes, it might be easier to use the CREATE statement to overwrite the old descriptor with a new one.

Altering a DBMS table might invalidate descriptor files that are based on the DBMS table, or it might cause these files to be out of date. If you re-create a table, add a new column to a table, or delete an existing column from a table, use the UPDATE statement to modify your descriptors so that they use the new information.

---

# Usage: ACCESS Procedure

---

## Using Descriptors with the ACCESS Procedure

---

### What Are Descriptors?

Descriptors work with the ACCESS procedure by providing information about DBMS objects to SAS, enabling you to access and update DBMS data from within a SAS session or program.

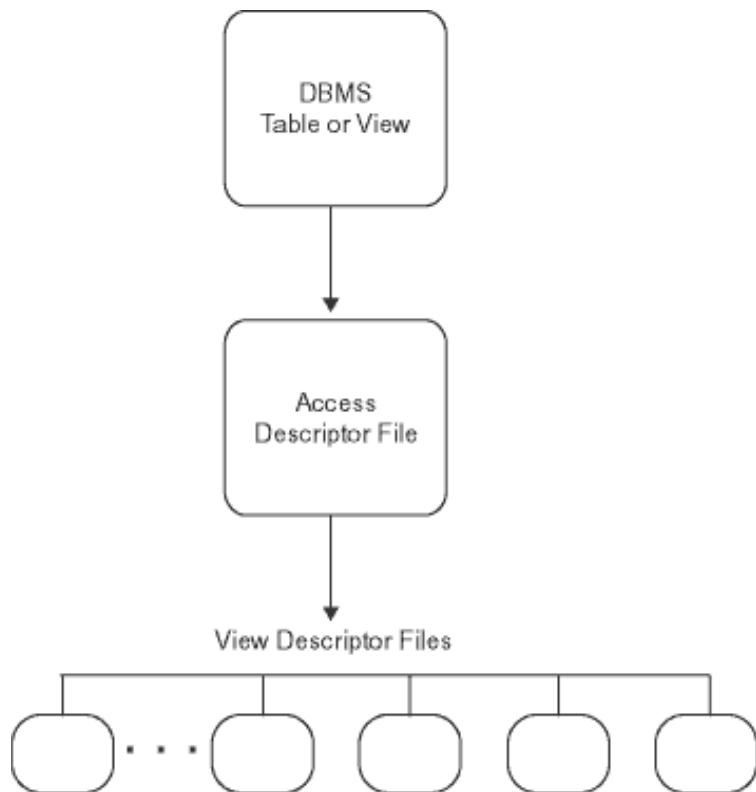
There are two types of descriptors, *access descriptors* and *view descriptors*. Access descriptors provide SAS with information about the structure and attributes of a DBMS table or view. In turn, an access descriptor is used to create one or more view descriptors, or SAS data views, of the DBMS data.

---

### Access Descriptors

Typically, each DBMS table or view has a single access descriptor that provides connection information, data type information, and names for databases, tables, and columns.

You use an access descriptor to create one or more view descriptors. When creating a view descriptor, you select the columns and specify criteria for the rows that you want to retrieve. The figure below illustrates the descriptor creation process. Note that an access descriptor, which contains the metadata of the DBMS table, must be created before view descriptors can be created.

**Figure A5.1** Creating an Access Descriptor and View Descriptors for a DBMS Table

## View Descriptors

You use a view descriptor in a SAS program much as you would any SAS data set. For example, you can specify a view descriptor in the DATA= statement of a SAS procedure or in the SET statement of a DATA step.

You can also use a view descriptor to copy DBMS data into a SAS data set, called *extracting* the data. When you need to use DBMS data in several procedures or DATA steps, you might use fewer resources by extracting the data into a SAS data set instead of repeatedly accessing the data directly.

The SAS/ACCESS interface view engine usually tries to pass WHERE conditions to the DBMS for processing. In most cases it is more efficient for a DBMS to process WHERE conditions than for SAS to do the processing.

## Accessing Data Sets and Descriptors

SAS lets you control access to SAS data sets and access descriptors by associating one or more SAS passwords with them. When you create an access descriptor, the connection information that you provide is stored in the access descriptor and in any view descriptors based on that access descriptor. The password is stored in an encrypted form. When these descriptors are accessed, the connection information that was stored is also used to access the DBMS table or view. To ensure data

security, you might want to change the protection on the descriptors to prevent others from seeing the connection information stored in the descriptors.

When you create or update view descriptors, you can use a SAS data set option after the ACCDESC= option to specify the access descriptor password, if one exists. In this case, you are *not* assigning a password to the view descriptor that is being created or updated. Instead, using the password grants you permission to use the access descriptor to create or update the view descriptor. Here is an example:

```
proc access dbms=sapase accdesc=adlib.customer (alter=rouge);
  create vlib.customer.view;
  select all;
run;
```

By specifying the ALTER level of password, you can read the AdLib.Customer access descriptor and create the Vlib.Customer view descriptor.

## Examples: ACCESS Procedure

### Example 1: Update an Access Descriptor

|           |                                                     |
|-----------|-----------------------------------------------------|
| Features: | Statements<br>PROC ACCESS<br>UPDATE<br>DROP<br>LIST |
|-----------|-----------------------------------------------------|

The following example updates an AdLib.Employ access descriptor on the Employees table in Oracle. The original access descriptor includes all columns in the table. The updated access descriptor omits the Salary and BirthDate columns. You can use the LIST statement to write all variables to the SAS log so that you can see the complete access descriptor before you update it.

```
proc access dbms=oracle ad=adlaib.employ;
  /* update access descriptor */
  update adlib.employ.access;
  drop salary birthdate;
  list all;
run;
```

### Example 2: Create a View Descriptor

|           |                                               |
|-----------|-----------------------------------------------|
| Features: | Statements<br>PROC ACCESS<br>CREATE<br>SELECT |
|-----------|-----------------------------------------------|

**FORMAT  
SUBSET**

---

Because SELECT and RESET are not supported when UPDATE is used, the view descriptor Vlib.Emp1204 must be re-created to omit the Salary and BirthDate columns.

Based on a previously updated access descriptor, ADLIB.EMPLOY, the following example re-creates the VLIB.EMP1204 view descriptor.

```
proc access dbms=oracle;
  /* re-create view descriptor */
  create vlib.emp1204.view;
  select empid hiredate dept jobcode gender
    lastname firstname middlename phone;
  format empid 6.
    jobcode 5.
    hiredate datetime9.;
  subset where jobcode=1204;
run;
```

# Chapter 43

## CV2VIEW Procedure

---

|                                                                               |             |
|-------------------------------------------------------------------------------|-------------|
| <b>Overview: CV2VIEW Procedure .....</b>                                      | <b>1443</b> |
| Overview: CV2VIEW Procedure .....                                             | 1443        |
| <b>Syntax: CV2VIEW Procedure .....</b>                                        | <b>1444</b> |
| PROC CV2VIEW Statement .....                                                  | 1445        |
| FROM_VIEW Statement .....                                                     | 1445        |
| FROM_LIBREF Statement .....                                                   | 1446        |
| REPLACE Statement .....                                                       | 1446        |
| SAVEAS Statement .....                                                        | 1447        |
| SUBMIT Statement .....                                                        | 1447        |
| TO_VIEW Statement .....                                                       | 1448        |
| TO_LIBREF Statement .....                                                     | 1448        |
| TYPE Statement .....                                                          | 1449        |
| <b>Examples: CV2VIEW Procedure .....</b>                                      | <b>1450</b> |
| Example 1: Converting an Individual View Descriptor .....                     | 1450        |
| Example 2: Converting a Library of View Descriptors for a Single DBMS .....   | 1451        |
| Example 3: Converting a Library of View Descriptors for All Supported DBMSs . | 1452        |

---

## Overview: CV2VIEW Procedure

---

### Overview: CV2VIEW Procedure

SAS still supports this legacy procedure, which converts into SQL views any SAS/ACCESS 32- and 64-bit view descriptors that were created prior to SAS 9.2. To access your relational DBMS data more directly, it is recommended that you convert your descriptors so that you can instead use the preferred method, the [SAS/ACCESS LIBNAME statement on page 127](#). The LIBNAME statement provides greater control over DBMS operations such as locking, spooling, and data type conversions. It can handle long field names; descriptors cannot handle long field

names. Also, SQL views are platform-independent; descriptors are not platform-independent.

If the descriptor that you want to convert is READ-, WRITE-, or ALTER-protected, those values are applied to the output SQL view. For security reasons, these values do not appear if you save the generated SQL to a file. The PASSWORD part of the LIBNAME statement is also not visible to prevent generated SQL statements from being submitted manually without modification.

## Syntax: CV2VIEW Procedure

See: [Appendix 2, “CV2VIEW Procedure,” on page 1443](#)

```
PROC CV2VIEW DBMS=dbms-name | ALL;
  FROM_LIBREF=libref.input-descriptor;
  FROM_VIEW=libref.input-view;
  TYPE=SQL | VIEW | ACCESS;
  TO_LIBREF=name-of-output-library;
  SAVEAS=external-file-name;
  SUBMIT;
  REPLACE=ALL | VIEW | FILE;
```

| Statement           | Task                                                                                               |
|---------------------|----------------------------------------------------------------------------------------------------|
| <b>PROC CV2VIEW</b> | Convert SAS/ACCESS view descriptors into SQL views                                                 |
| <b>FROM_LIBREF</b>  | Specify the library containing the view descriptors or access descriptors that you want to convert |
| <b>FROM_VIEW</b>    | Specify the view descriptor or access descriptor that you want to convert                          |
| <b>REPLACE</b>      | Specify whether existing views and files are replaced                                              |
| <b>SAVEAS</b>       | Save the generated PROC SQL statements to a file                                                   |
| <b>SUBMIT</b>       | Submit the generated PROC SQL statements when you specify the SAVEAS statement                     |
| <b>TO_VIEW</b>      | Specify the name of the new (converted) SQL view                                                   |
| <b>TO_LIBREF</b>    | Specify the library that contains the new (converted) SQL views                                    |
| <b>TYPE</b>         | Specify what type of conversion should occur                                                       |

---

## PROC CV2VIEW Statement

Converts SAS/ACCESS view descriptors into SQL views

---

### Syntax

**PROC CV2VIEW DBMS=*dbms-name* | ALL;**

### Required Arguments

***dbms-name***

specifies the name of a supported database from which you want to obtain descriptors. Valid values for *dbms-name* are **DB2**, **Oracle**, and **SAPASE**.

**ALL**

specifies that you want the descriptors from all supported databases.

---

## FROM\_VIEW Statement

Specifies the name of the view descriptor or access descriptor that you want to convert

Restriction: If you specify DBMS=ALL, you cannot use the FROM\_VIEW statement.

Requirements: You must specify either the FROM\_VIEW statement or the FROM\_LIBREF statement. FROM\_VIEW and TO\_VIEW statements are always used together.

---

### Syntax

**FROM\_VIEW=*libref.input-descriptor*,**

### Required Arguments

***libref***

specifies the libref that contains the view descriptor or access descriptor that you want to convert.

***input-descriptor***

specifies the view descriptor or access descriptor that you want to convert.

---

## FROM\_LIBREF Statement

Specifies the library that contains the view descriptors or access descriptors that you want to convert

Requirements: You must specify either the FROM\_VIEW statement or the FROM\_LIBREF statement. FROM\_LIBREF and TO\_LIBREF statements are always used together.

---

### Syntax

**FROM\_LIBREF=** *input-library*;

### Required Argument

***input-library***

specifies a previously assigned library that contains the view descriptors or access descriptors that you want to convert. All descriptors that are in the specified library and that access data in the specified DBMS are converted into SQL views. If you specify DBMS=ALL, all descriptors that are in the specified library and that access any supported DBMS are converted.

---

## REPLACE Statement

Specifies whether existing views and files are replaced

---

### Syntax

**REPLACE=**ALL | FILE | VIEW;

### Required Arguments

**ALL**

replaces the TO\_VIEW file if it already exists and replaces the SAVEAS file if it already exists.

**FILE**

replaces the SAVEAS file if it already exists. If the file already exists, and if REPLACE=FILE or REPLACE=ALL is not specified, the generated PROC SQL code is appended to the file.

**VIEW**

replaces the TO\_VIEW file if it already exists.

---

## SAVEAS Statement

Saves the generated PROC SQL statements to a file

Interaction: If you specify the SAVEAS statement, the generated SQL is not automatically submitted, so you must use the [SUBMIT statement on page 1447](#).

---

### Syntax

**SAVEAS=***external-file-name*;

### Required Argument

***external-file-name***

lets you save the PROC SQL statements that are generated by PROC CV2VIEW to an external file. You can modify this file and submit it on another platform.

---

### Details

PROC CV2VIEW inserts comments in the generated SQL to replace any statements that contain passwords. For example, if a view descriptor is READ-, WRITE-, or ALTER-protected, the output view has the same level of security. However, the file that contains the SQL statements does not show password values. The password in the LIBNAME statement also does not show password values.

---

## SUBMIT Statement

Causes PROC CV2VIEW to submit the generated PROC SQL statements when you specify the SAVEAS statement

Tip: If you do not use the [SAVEAS statement on page 1447](#), PROC CV2VIEW automatically submits the generated SQL. So you do not need to specify the SUBMIT statement.

---

### Syntax

**SUBMIT;**

---

## TO\_VIEW Statement

Specifies the name of the new (converted) SQL view

Restriction: If you specify DBMS=ALL, you cannot use the TO\_VIEW statement.

Requirements: You must specify either the TO\_VIEW statement or the TO\_LIBREF statement. FROM\_VIEW and TO\_VIEW statements are always used together.

Interaction: Use the [REPLACE statement on page 1446](#) to control whether the output file is overwritten or appended if it already exists.

---

## Syntax

**TO\_VIEW=***libref.output-view*;

### Required Arguments

***libref***

specifies the libref where you want to store the new SQL view.

***output-view***

specifies the name for the new SQL view that you want to create.

---

## TO\_LIBREF Statement

Specifies the library that contains the new (converted) SQL views

Requirements: You must specify either the TO\_VIEW statement or the TO\_LIBREF statement. The FROM\_LIBREF and TO\_LIBREF statements are always used together.

Interaction: Use the [REPLACE statement on page 1446](#) if a file with the name of one of your output views already exists. If a file with the name of one of your output views already exists and you do not specify the REPLACE statement, PROC CV2VIEW does not convert that view.

---

## Syntax

**TO\_LIBREF=***output-library*;

## Required Argument

### ***output-library***

specifies the name of a previously assigned library where you want to store the new SQL views.

---

## Details

The names of the input view descriptors or access descriptors are used as the output view names. In order to individually name your output views, use the [FROM\\_VIEW statement on page 1445](#) and the [TO\\_VIEW statement on page 1448](#).

---

## TYPE Statement

Specifies what type of conversion should occur

---

## Syntax

**TYPE=SQL | VIEW | ACCESS;**

## Required Arguments

### **SQL**

specifies that PROC CV2VIEW converts descriptors to SQL views. This is the default behavior.

### **VIEW**

specifies that PROC CV2VIEW converts descriptors to native view descriptor format. It is most useful in the 32-bit to 64-bit case. It does not convert view descriptors across different operating systems.

### **ACCESS**

specifies that PROC CV2VIEW converts access descriptors to native access descriptor format. It is most useful in the 32-bit to 64-bit case. It does not convert access descriptors across different operating systems.

---

## Details

If TYPE=VIEW or TYPE=ACCESS, then SAVEAS, SUBMIT, and REPLACE or REPLACE\_FILE are not valid options.

# Examples: CV2VIEW Procedure

## Example 1: Converting an Individual View Descriptor

| Features: | Statements   |
|-----------|--------------|
|           | PROC CV2VIEW |
|           | FROM_VIEW    |
|           | TO_VIEW      |
|           | SAVEAS       |
|           | SUBMIT       |
|           | REPLACE      |

In this example, PROC CV2VIEW converts the MYVIEW view descriptor to the SQL view NEWVIEW. When you use ALTER, READ, and WRITE, the MYVIEW view descriptor is protected again alteration, reading, and writing. The PROC SQL statements that PROC CV2VIEW generates are submitted and saved to an external file named SQL.SAS.

## Details

The REPLACE FILE statement causes an existing file named SQL.SAS to be overwritten. Without this statement, the text would be appended to SQL.SAS if the user has the appropriate privileges.

The LABEL value of EMPLINFO is the name of the underlying database table that is referenced by the view descriptor.

If the underlying DBMS is Oracle or DB2, the CV2VIEW procedure adds the PRESERVE\_TAB\_NAMES= option to the embedded LIBNAME statement. You can then use CV2VIEW to access those tables with mixed-case or embedded-blank table names.

---

**Note:** This SQL syntax fails if you try to submit it because the PW field of the LIBNAME statement is replaced with a comment in order to protect the password. The ALTER, READ, and WRITE protection is commented out for the same reason. You can add the passwords to the code and then submit the SQL to re-create the view.

---

```
libname input '/username/descriptors/';
libname output '/username/sqlviews/';
```

```

proc cv2view dbms=oracle;
from_view = input.myview (alter=apwd);
to_view = output.newview;
saveas = '/username/vsql/sql.sas';
submit;
replace file;
run;

```

PROC CV2VIEW generates these PROC SQL statements.

```

/* SOURCE DESCRIPTOR: MYVIEW */
PROC SQL DQUOTE=ANSI;
CREATE VIEW OUTPUT.NEWVIEW
(
/* READ= */
/* WRITE= */
/* ALTER= */
LABEL=EMPLINFO
)
AS SELECT
    "EMPLOYEE" AS EMPLOYEE INFORMAT= 5.0 FORMAT= 5.0
        LABEL= 'EMPLOYEE' ,
    "LASTNAME" AS LASTNAME INFORMAT= $10. FORMAT= $10.
        LABEL= 'LASTNAME' ,
    "SEX" AS SEX INFORMAT= $6. FORMAT= $6.
        LABEL= 'SEX' ,
    "STATUS" AS STATUS INFORMAT= $9. FORMAT= $9.
        LABEL= 'STATUS' ,
    "DEPARTMENT" AS DEPARTME INFORMAT= 7.0 FORMAT= 7.0
        LABEL= 'DEPARTMENT' ,
    "CITYSTATE" AS CITYSTAT INFORMAT= $15. FORMAT= $15.
        LABEL= 'CITYSTATE'
FROM _CVLIB_."EMPLINFO"
USING LIBNAME _CVLIB_
Oracle
/* PW= */
USER=ordevxx PATH=OracleV8 PRESERVE_TAB_NAMES=YES;
QUIT;

```

## Example 2: Converting a Library of View Descriptors for a Single DBMS

| Features: | Statements   |
|-----------|--------------|
|           | PROC CV2VIEW |
|           | FROM_LIBREF  |
|           | TO_LIBREF    |
|           | SAVEAS       |
|           | SUBMIT       |

In this example, PROC CV2VIEW converts all Oracle view descriptors in the input library into SQL views. If an error occurs during the conversion of a view descriptor, the procedure moves to the next view. The PROC SQL statements that PROC

CV2VIEW generates are both submitted and saved to an external file named SQL.SAS.

The SAVEAS statement causes all generated SQL for all Oracle view descriptors to be stored in the MANYVIEW.SAS file. If the underlying DBMS is Oracle or DB2, the CV2VIEW procedure adds the PRESERVE\_TAB\_NAMES= option to the embedded LIBNAME statement. You can then use CV2VIEW to access those tables with mixed-case or embedded-blank table names.

```
libname input '/username/descriptors/';
libname output '/username/sqlviews/';
proc cv2view dbms=oracle;
from_libref = input;
to_libref = output;
saveas = '/username/vsql/manyview.sas';
submit;
run;
```

PROC CV2VIEW generates these PROC SQL statements for one of the views.

```
/* SOURCE DESCRIPTOR: PPCV2R */
PROC SQL DQUOTE=ANSI;
CREATE VIEW OUTPUT.PPCV2R
(
LABEL=EMPLOYEES
)
AS SELECT
"EMPID"      " AS EMPID INFORMAT= BEST22. FORMAT= BEST22.
               LABEL= 'EMPID'      ,
"HIREDATE"   " AS HIREDATE INFORMAT= DATETIME16. FORMAT= DATETIME16.
               LABEL= 'HIREDATE'   ,
"JOBCODE"    " AS JOBCODE INFORMAT= BEST22. FORMAT= BEST22.
               LABEL= 'JOBCODE'    ,
"SEX"        " AS SEX INFORMAT= $1. FORMAT= $1.
               LABEL= 'SEX'        '
FROM _CVLIB_."EMPLOYEES" (
SASDATEFMT = ( "HIREDATE"= DATETIME16. ) )
USING LIBNAME _CVLIB_
Oracle
/* PW= */
USER=ordevxx PATH=OracleV8 PRESERVE_TAB_NAMES=YES;
QUIT;
```

---

## Example 3: Converting a Library of View Descriptors for All Supported DBMSs

Features:      PROC CV2VIEW option  
                  DBMS=ALL

Statements

```
PROC CV2VIEW
FROM_LIBREF
TO_LIBREF
```

In this example, PROC CV2VIEW converts all view descriptors that are in the input library and that access data in any supported DBMS. If an error occurs during the conversion of a view descriptor, the procedure then moves to the next view. The PROC SQL statements that PROC CV2VIEW generates are automatically submitted but are not saved to an external file because the [SAVEAS statement](#) on [page 1447](#) is not used.

```
libname input '/username/descriptors/';
libname output '/username/sqlviews/';

proc cv2view dbms=all;
from_libref = input;
to_libref = output;
run;
```



# Chapter 44

## DBLOAD Procedure

---

|                                                             |             |
|-------------------------------------------------------------|-------------|
| <i>Overview: DBLOAD Procedure</i> . . . . .                 | <b>1455</b> |
| Sending Data from SAS to a DBMS . . . . .                   | 1456        |
| Properties of the DBLOAD Procedure . . . . .                | 1456        |
| About DBLOAD Procedure Statements . . . . .                 | 1456        |
| <i>Concepts: DBLOAD Procedure</i> . . . . .                 | <b>1457</b> |
| DBMS Specifics: DBLOAD Procedure . . . . .                  | 1457        |
| <i>Syntax: DBLOAD Procedure</i> . . . . .                   | <b>1458</b> |
| PROC DBLOAD Statement . . . . .                             | 1459        |
| Database Connection Statement . . . . .                     | 1460        |
| ACCDESC Statement . . . . .                                 | 1460        |
| COMMIT Statement . . . . .                                  | 1461        |
| DELETE Statement . . . . .                                  | 1461        |
| ERRLIMIT Statement . . . . .                                | 1462        |
| LABEL Statement . . . . .                                   | 1462        |
| LIMIT Statement . . . . .                                   | 1463        |
| LIST Statement . . . . .                                    | 1463        |
| LOAD Statement . . . . .                                    | 1464        |
| NULLS Statement . . . . .                                   | 1464        |
| QUIT Statement . . . . .                                    | 1465        |
| RENAME Statement . . . . .                                  | 1466        |
| RESET Statement . . . . .                                   | 1467        |
| SQL Statement . . . . .                                     | 1467        |
| TABLE Statement . . . . .                                   | 1468        |
| TYPE Statement . . . . .                                    | 1469        |
| WHERE Statement . . . . .                                   | 1469        |
| <i>Example: Append a Data Set to a DBMS Table</i> . . . . . | <b>1470</b> |

---

# Overview: DBLOAD Procedure

---

## Sending Data from SAS to a DBMS

SAS still supports this legacy procedure. However, to send data from SAS to a DBMS more directly, it is recommended that you use the [SAS/ACCESS LIBNAME statement on page 127](#) or the [SQL pass-through facility on page 690](#). To determine whether this feature is available in your environment for your interface, see [SAS/ACCESS Features by Host on page 96](#).

---

## Properties of the DBLOAD Procedure

Along with the ACCESS procedure and an interface view engine, the DBLOAD procedure creates an interface between SAS and data in other vendors' databases.

The DBLOAD procedure enables you to create and load a DBMS table, append rows to an existing table, and submit non-query DBMS-specific SQL statements to the DBMS for processing. The procedure constructs DBMS-specific SQL statements to create and load, or append, to a DBMS table by using one of these items:

- a SAS data set
- an SQL view or DATA step view
- a view descriptor that was created with the SAS/ACCESS interface to your DBMS or with another SAS/ACCESS interface product
- another DBMS table referenced by a SAS libref that was created with the SAS/ACCESS LIBNAME statement.

The DBLOAD procedure associates each SAS variable with a DBMS column and assigns a default name and data type to each column. It also specifies whether each column accepts NULL values. You can use the default information or change it as necessary. When you are finished customizing the columns, the procedure creates the DBMS table and loads or appends the input data.

---

## About DBLOAD Procedure Statements

This procedure has several types of statements:

- *database connection statements* for connecting to your DBMS (see DBMS-specific information in this document for your SAS/ACCESS interface)
- *creating and loading statements*: [LOAD](#) and [RUN](#)

- *table and editing statements:* ACCDESC, COMMIT, DELETE, ERRLIMIT, LABEL, LIMIT, LIST, NULLS, QUIT, RENAME, RESET, SQL, TABLE, TYPE, WHERE

This table summarizes PROC DBLOAD options and statements that are required to accomplish common tasks.

**Table A7.1 Statement Sequence for Accomplishing Common Tasks with the DBLOAD Procedure**

| Task                                                                                                            | Options and Statements to Use                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Create and load a DBMS table (LOAD must appear before RUN to create and load a table or append data to a table) | <b>PROC DBLOAD</b><br><i>statement-options;</i><br><i>database-connection-options;</i><br><b>TABLE=&lt;'&gt;table-name&lt;'&gt;;</b><br><b>LOAD;</b><br><b>RUN;</b> |
| Submit a dynamic, non-query DBMS-SQL statement to DBMS (without creating a table)                               | <b>PROC DBLOAD</b><br><i>statement-options;</i><br><i>database-connection-options;</i><br><b>SQL DBMS-specific-SQL-statements;</b><br><b>RUN;</b>                   |

## Concepts: DBLOAD Procedure

### DBMS Specifics: DBLOAD Procedure

These SAS/ACCESS interfaces support this procedure:

- DB2 under UNIX and PC Hosts
- DB2 under z/OS
- Microsoft SQL Server
- ODBC
- Oracle
- SAP ASE

# Syntax: DBLOAD Procedure

See: DB2 under UNIX and PC Hosts, DB2 under z/OS, Microsoft SQL Server, ODBC, Oracle, SAP ASE

```
PROC DBLOAD <options>;
   database connection statements;
   TABLE=<'>table-name <'>;
   ACCDESC=<libref.>access-descriptor;
   COMMIT=commit-frequency;
   DELETE variable-identifier-1
      <...variable-identifier-n>;
   ERRLIMIT=error-limit;
   LABEL;
   LIMIT=load-limit;
   LIST <ALL | FIELD | <'>variable-identifier<'>>;
   NULLS variable-identifier-1 = Y | N
      <...variable-identifier-n = Y | N>;
   QUIT;
   RENAME variable-identifier-1 = <'>column-name-1<'>
      <...variable-identifier-n = <'>column-name-n<'>>;
   RESET ALL | variable-identifier-1<...variable-identifier-n>;
   SQL DBMS-specific SQL-statement;
   TYPE variable-identifier-1 = 'column-type-1'
      <...variable-identifier-n = 'column-type-n'>;
   WHERE SAS-where-expression;
   LOAD;
RUN;
```

| Statement                  | Task                                                            |
|----------------------------|-----------------------------------------------------------------|
| <b>PROC DBLOAD</b>         | Send data from SAS to a DBMS                                    |
| <b>Database Connection</b> | Provide DBMS connection information                             |
| <b>ACCDESC</b>             | Create an access descriptor based on the new DBMS table         |
| <b>COMMIT</b>              | Issue a commit or save rows after a specified number of inserts |
| <b>DELETE</b>              | Specify variables that should not be loaded into the new table  |
| <b>ERRLIMIT</b>            | Stop the loading of data after a specified number of errors     |

| Statement | Task                                                      |
|-----------|-----------------------------------------------------------|
| LABEL     | Cause DBMS column names to default to SAS labels          |
| LIMIT     | Limit the number of observations that are loaded          |
| LIST      | List information about the variables to be loaded         |
| LOAD      | Create and load the new DBMS table                        |
| NULLS     | Specify whether DBMS columns accept NULL values           |
| QUIT      | Terminate the procedure                                   |
| RENAME    | Rename DBMS columns                                       |
| RESET     | Reset column names and data types to their default values |
| SQL       | Submit a DBMS-specific SQL statement to the DBMS          |
| TABLE     | Name the DBMS table to be created and loaded              |
| TYPE      | Change default DBMS data types in the new table           |
| WHERE     | Load a subset of data into the new table                  |

---

## PROC DBLOAD Statement

Sends data from SAS to a DBMS

---

## Syntax

**PROC DBLOAD <options>;**

### Optional Arguments

**DBMS=***database-management-system*

specifies which database management system you want to access. This DBMS-specific option is required. See the DBMS-specific reference in this document for details about your DBMS.

**DATA=<libref.>***SAS-data-set*

specifies the input data set. You can retrieve data from a SAS data set, an SQL view, a DATA step view, a SAS/ACCESS view descriptor, or another DBMS table to which a SAS/ACCESS libref points. If the SAS data set is permanent, you must use its two-level name, *libref.SAS-data-set*. If you omit the DATA= option, the default is the last SAS data set that was created.

**APPEND**

appends data to an existing DBMS table that you identify by using the TABLE statement. When you specify APPEND, the input data that you specify with the DATA= option is inserted into the existing DBMS table. Your input data can be a SAS data set, SQL view, or SAS/ACCESS view (view descriptor).

**CAUTION**

**When you use APPEND, you must ensure that your input data corresponds exactly to the columns in the DBMS table. If your input data does not include values for all columns in the DBMS table, you might corrupt your DBMS table by inserting data into the wrong columns. Use the COMMIT, ERRLIMIT, and LIMIT statements to help safeguard against data corruption. Use the DELETE and RENAME statements to drop and rename SAS input variables that do not have corresponding DBMS columns.**

---

You can use all PROC DBLOAD statements and options with APPEND except NULLS and TYPE, which have no effect. The LOAD statement is required.

The following example appends new employee data from the NEWEMP SAS data set to the EMPLOYEES DBMS table. The COMMIT statement causes a DBMS commit to be issued after every 100 rows are inserted. The ERRLIMIT statement causes processing to stop after five errors occur.

```
proc dbload dbms=oracle data=newemp append;
  user=myusr1; password=mypwd1; path='mysrv1';
  table=employees; commit=100; errlimit=5; load;
run;
```

By omitting the APPEND option from the DBLOAD statement, you can use the PROC DBLOAD SQL statements to create a DBMS table and append to it in the same PROC DBLOAD step.

---

## Database Connection Statement

Provides DBMS connection information

|              |                                                                                                                                                                                                                                                                         |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Interaction: | These statements are used to connect to your DBMS. They vary based on the SAS/ACCESS interface that you are using (for example, include USER=, PASSWORD=, and DATABASE=). See the DBMS-specific reference in this document for details about your SAS/ACCESS interface. |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

## Syntax

*database-connection-statements*

---

## ACCDESC Statement

Creates an access descriptor based on the new DBMS table that you are creating and loading

- Requirement: You must specify an access descriptor if it does not already exist.
- Tip: If you specify ACCDESC, the access descriptor is automatically created after the new table is created and loaded.

## Syntax

**ACCDESC=<libref.>access-descriptor;**

## COMMIT Statement

Issues a commit or saves rows after a specified number of inserts

- Default: 1000
- Requirement: The *commit-frequency* argument must be a nonnegative integer.
- Interaction: If you omit the COMMIT statement, a commit is issued or a group of rows is saved after each 1,000 rows are inserted and after the last row is inserted.
- Note: The COMMIT statement issues a commit (generates a DBMS-specific SQL COMMIT statement) after the specified number of rows has been inserted.
- Tip: Using this statement might improve performance by releasing DBMS resources each time the specified number of rows has been inserted.

## Syntax

**COMMIT=***commit-frequency*;

## Required Argument

### *commit-frequency*

specifies the number of inserts before committing rows.

## DELETE Statement

drops the specified SAS variables before the DBMS table is created

- Tip: When you drop a variable, the positional equivalents of the variables do not change. For example, if you drop the second variable, the third variable is still referenced by the number 3, not 2. If you drop more than one variable, separate the identifiers with spaces, not commas.

---

## Syntax

**DELETE** *variable-identifier-1* <...*variable-identifier-n*>;

### Required Argument

***variable-identifier***

specifies either the SAS variable name or the positional equivalent from the LIST statement. The positional equivalent is the number that represents the variable's place in the data set. For example, if you want to drop the third variable, submit this statement: `delete 3;`

---

## ERRLIMIT Statement

Stops the loading of data after a specified number of errors

Default: 100 (see the DBMS-specific details for possible exceptions)

Requirement: The *error-limit* argument must be a nonnegative integer.

Interaction: The ERRLIMIT statement defaults to 10 when used with APPEND.

Tip: To allow an unlimited number of DBMS SQL errors to occur, specify `ERRLIMIT=0`.

---

## Syntax

**ERRLIMIT=***error-limit*;

### Required Argument

***error-limit***

specifies the number of DBMS SQL errors that are allowed before SAS/ACCESS stops loading data.

Errors include observations that fail to be inserted and commits that fail to execute. If the SQL CREATE TABLE statement that the procedure generates fails, the procedure ends.

---

## LABEL Statement

Causes DBMS column names to default to SAS variable labels when the new table is created

Default: DBMS column names default to SAS variable names

Requirement: You must use the RESET statement after the LABEL statement for the LABEL statement to take effect.

Interaction: RESET

Note: If a SAS variable has no label, the variable name is used. If the label is too long to be a valid DBMS column name, the label is truncated.

## Syntax

**LABEL;**

## LIMIT Statement

Limits the number of observations that can be loaded into a new DBMS table

Default: 5000

Requirement: The *load-limit* argument must be a nonnegative integer.

Tip: To load all observations from your input data set, specify LIMIT=0.

## Syntax

**LIMIT=***load-limit*;

## Required Argument

### *load-limit*

specifies the number of observations to load into a new DBMS table.

## LIST Statement

Lists information about some or all of the SAS variables to be loaded into the new DBMS table

Default: ALL

Note: By default, the list is sent to the SAS log.

Tip: You can specify LIST as many times as you want while creating a DBMS table. Specify LIST before the LOAD statement to see the entire table.

## Syntax

**LIST <ALL | FIELD | *variable-identifier*>;**

## Optional Arguments

### **ALL**

lists information about all variables in the input SAS data set, despite whether those variables are selected for the load.

### **FIELD**

lists information about only the input SAS variables that are selected for the load.

### ***variable-identifier***

lists information about only the specified variable. The *variable-identifier* argument can be either the SAS variable name or the positional equivalent. The positional equivalent is the number that represents the variable's position in the data set. For example, if you want to list information for the column associated with the third SAS variable, submit this statement: `list 3;`

## LOAD Statement

Creates and loads the new DBMS table

- Restriction: in the DBLOAD procedure (required statement for loading or appending data)
- Requirements: This statement is required to create and load a new DBMS table or to append data to an existing table.  
When you create and load a DBMS table, you must place statements or groups of statements in a certain order after the PROC DBLOAD statement and its options, as listed in [Statement Sequence for Accomplishing Common Tasks on page 1457](#).
- Note: The LOAD statement informs the DBLOAD procedure to execute the action that you request, including loading or appending data.
- Example: This example creates the SummerTemps table in Oracle based on the DLib.TempEmps data file.

```
proc dbload dbms=oracle data=dlib.tempemps;
  user=myusr1; password=mypwd1; path='mysrv1';
  table=summertemps; rename firstnam=firstname middlena=middlename;
  type hiredate 'date' empid 'number(6,0)' familyid 'number(6,0)';
  nulls 1=n; list; load;
run;
```

## Syntax

**LOAD;**

## NULLS Statement

Specifies whether DBMS columns accept NULL values

- Default: Y

- Interaction: Some DBMSs have three valid values for this statement, Y, N, and D. See the DBMS-specific reference in this document for details about your DBMS.
- Tip: If you omit the NULLS statement, the DBMS default action occurs. You can list as many variables as you want in one NULLS statement. If you have previously specified a column as NULLS=N, you can use the NULLS statement to redefine it to accept NULL values.
- See: [ERRLIMIT on page 1462](#)

## Syntax

**NULLS** *variable-identifier-1* = Y | N <...*variable-identifier-n* = Y | N>;

## Required Argument

### *variable-identifier*

specifies either the SAS variable name or the positional equivalent from the LIST statement. The positional equivalent is the number that represents the variable's place in the data set. For example, submit nulls 3=y; if you want the column that is associated with the third SAS variable to accept NULL values.

## Details

The NULLS statement specifies whether the DBMS columns that are associated with the listed input SAS variables allow NULL values. Specify Y to accept NULL values. Specify N to reject NULL values and to require data in that column.

If you specify N for a numeric column, no observations that contain missing values in the corresponding SAS variable are loaded into the table. A message is written to the SAS log, and the current error count increases by one for each observation that is not loaded.

If a character column contains blanks (the SAS missing value) and you have specified N for the DBMS column, blanks are inserted. If you specify Y, NULL values are inserted.

## QUIT Statement

Terminates the procedure without further processing

- Restriction: Valid in the DBLOAD procedure (control statement)

## Syntax

**QUIT;**

---

## RENAME Statement

Renames DBMS columns

- Requirements: The *column-name* argument must be a valid DBMS column name. If the column name includes lowercase characters, special characters, or national characters, you must enclose the column name in single or double quotation marks. If no quotation marks are used, the DBMS column name is created in uppercase. Use `rename 3='employeeName'` syntax to preserve case.
- Interactions: DELETE, LABEL, RESET  
The RENAME statement overrides the LABEL statement for columns that are renamed.
- Notes: This statement changes the names of the DBMS columns that are associated with the listed SAS variables.  
If you omit the RENAME statement, all DBMS column names default to the corresponding SAS variable names unless you specify the LABEL statement.
- Tip: You can list as many variables as you want in one RENAME statement. COLUMN is an alias for the RENAME statement.
- Example: You can use this statement to include variables that you have previously deleted, as shown in this example:

```
delete 3;
rename 3=empname;
```

The DELETE statement drops the third variable. The RENAME statement includes the third variable and assigns the name EMPNAME and the default column type to it.

---

## Syntax

```
RENAME variable-identifier-1 = <'>column-name-1<'>
<...variable-identifier-n = <'>column-name-n<'>>;
```

## Required Arguments

### *variable-identifier*

specifies either the SAS variable name or the positional equivalent from the LIST statement. The positional equivalent is the number that represents where to place the variable in the data set. For example, submit `rename 3=employeeName`; if you want to rename the column associated with the third SAS variable.

### *column-name*

indicates a new column name for the specified variable.

---

# RESET Statement

Resets column names and data types to their default values

Requirement: You must use the RESET statement after the LABEL statement for the LABEL statement to take effect.

Interaction: DELETE, LABEL, RENAME, TYPE

---

## Syntax

**RESET ALL | *variable-identifier-1* <...*variable-identifier-n*>;**

### Required Argument

#### ***variable-identifier***

specifies either the SAS variable name or the positional equivalent from the LIST statement. The positional equivalent is the number that represents the variable's place in the data set. For example, submit `reset 3;` if you want to reset the column associated with the third SAS variable.

---

## Details

The RESET statement resets columns that are associated with the listed SAS variables to default values for the DBMS column name, column data type, and ability to accept NULL values. If you specify ALL, all columns are reset to their default values, and any dropped columns are restored with their default values. Here are the default values.

#### column name

defaults to the SAS variable name, or to the SAS variable label (if you have used the LABEL statement).

#### column type

is generated from the SAS variable format.

#### nulls

uses the DBMS default value.

---

# SQL Statement

Submits a dynamic, nonquery, DBMS-specific SQL statement to the DBMS

- Restriction:** You cannot create a DBMS table and reference it in your DBMS-specific SQL statements within the same PROC DBLOAD step. The new table is not created until the RUN statement is processed.
- Requirements:** You must enter the keyword SQL before each DBMS-specific SQL statement that you submit.  
You must use DBMS-specific SQL object names and syntax in the DBLOAD SQL statement.
- Note:** You can use the DBLOAD statement to submit these DBMS-specific SQL statements, despite whether you create and load a DBMS table.
- Tip:** To submit dynamic, non-query DBMS-specific SQL statements to the DBMS without creating a DBMS table, use the DBMS= option, any database connection statements, and the SQL statement.
- Example:** This example grants UPDATE privileges to user MARURI on the DB2 SasDemo.Orders table.

```
proc dbload dbms=db2;
  in sample;
  sql grant update on sasdemo.orders to maruri;
run;
```

## Syntax

**SQL** *DBMS-specific-SQL-statement*;

## Required Argument

### **DBMS-specific-SQL-statement**

specifies any valid dynamic DBMS-specific SQL statement except the SELECT statement. However, you can enter a SELECT statement as a substatement within another statement, such as in a CREATE VIEW statement.

## TABLE Statement

specifies the name of the DBMS table to be created and loaded into a DBMS database

- Restriction:** do not use this statement when you are submitting dynamic DBMS-specific SQL statements to the DBMS without creating and loading a table.
- Requirements:** This statement is required when you create and load or append to a DBMS table. It must follow other database connection statements such as DATABASE= or USER=.  
You must specify a table name that does not already exist. If a table by that name exists, an error message is written to the SAS log, and the table specified in this statement is not loaded.

---

## Syntax

**TABLE=**<'>*DBMS-specific-syntax*<'>;

### Required Argument

***DBMS-specific-syntax***

specifies a valid table name for the DBMS. (See the DBMS-specific reference in this document for the syntax for your DBMS.)

If your table name contains lowercase characters, special characters, or national characters, it must be enclosed in quotation marks.

---

## TYPE Statement

Changes default DBMS data types in a new table

**Tip:** If you omit the TYPE statement, the column data types are generated with default DBMS data types that are based on the SAS variable formats. You can change as many data types as you want in one TYPE statement. See the DBMS-specific reference in this document for a complete list of default conversion data types for the DBLOAD procedure for your DBMS.

---

## Syntax

**TYPE** *variable-identifier-1* = '*column-type-1*'  
<...*variable-identifier-n* = '*column-type-n*'>;

### Required Arguments

***variable-identifier***

specifies either the SAS variable name or the positional equivalent from the LIST statement. The positional equivalent is the number that represents the variable's place in the data set. For example, submit type 3='char(17)'; if you want to change the data type of the DBMS column associated with the third SAS variable.

***column-type***

specifies a valid data type for the DBMS and must be enclosed in quotation marks.

---

## WHERE Statement

Loads a subset of data observations into the new table

See: [SAS System Options: Reference](#) (syntax)

Example: This example loads only the observations in which the COUNTRY SAS variable has the value of BRAZIL:

```
where country='Brazil';
```

## Syntax

**WHERE** *SAS-where-expression*;

## Required Argument

### **SAS-where-expression**

specifies a valid SAS WHERE clause that uses SAS variable names, not DBMS column names, as specified in the input data set.

## Example: Append a Data Set to a DBMS Table DBLOAD Procedure

### Example 1: Append a Data Set to a DBMS Table

Features: PROC DBLOAD option  
APPEND

Statements  
PROC DBLOAD  
TABLE  
COMMIT  
ERRLIMIT

Database connection statements  
USER  
password  
PATH

By omitting the APPEND option from the DBLOAD statement, you can use the PROC DBLOAD SQL statements to create a DBMS table and append to it in the same PROC DBLOAD step.

The following example appends new employee data from the NewEmp SAS data set to the Employees DBMS table. The COMMIT statement causes a DBMS commit to be issued after every 100 rows are inserted. The ERRLIMIT statement causes processing to stop after 10 errors occur.

```
proc dblload dbms=oracle data=newemp append;
  user=myusr1; password=mypwd1; path='mysrv1';
  table=employees; commit=100; errlimit=10; load;
run;
```

