SAP BusinessObjects™

Supplement for SAP

■ SAP BusinessObjects Data Services 4.1 Support Package 1 (14.1.1.0)

2012-12-12

SAP

Contents

Introduction

This guide contains information about how to use Data Services with SAP applications and SAP NetWeaver Business Warehouse. It is organized to complement the information in the *Data Services Designer Guide*, and the reference section complements the information in the *Data Services Reference Guide*.

1.1 Terms

The following table describes SAP terminology for this release.

Term	Abbreviation	Description
SAP Applications	Not applicable	Applications are the main building blocks of the SAP solution portfolios for industries. They provide the software foundation with which organizations address their business issues. SAP delivers the following types of applications: • General-purpose applications: These include applications provided within SAP Business Suite software such as the SAP Customer Relationship Management application and the SAP ERP application. • Industry-specific applications: These applications perform targeted, industry-specific business functions. Examples are the SAP Apparel and Footwear application for the consumer products industry and the SAP Reinsurance Management application for the insurance industry. SAP R/3 software is available today as SAP ERP. Therefore, SAP R/3 software functionality falls under the category of SAP Applications.
SAP Master Data Services	Not applicable	A solution powered by SAP HANA that provides enterprises with instant access to a complete view of their entire customer's information and relationships—data that is seamlessly consolidated and enriched from heterogeneous sources across the enterprise.
SAP NetWeaver technology platform	Not applicable	Helps organizations perform a wide variety of IT processes such as user productivity enablement, information management, and business process management. Includes several components and tools
SAP NetWeaver Application Server component	SAP NetWeaver AS	Supports platform-independent Web services, business applications, and standards-based development. This software enables you to leverage existing technology assets for Web-services-oriented solutions. It is a component of the SAP NetWeaver technology platform. Formerly referred to as the SAP Web Application Server or Basis.
SAP NetWeaver Business Warehouse component	SAP NetWeaver Business Warehouse	Enables you to integrate data from across the enterprise and transform it into practical, timely business information to drive sound decision-making. It is a component of the SAP NetWeaver technology platform.

1.2 SAP interfaces

Data Services includes the following SAP interfaces:

- SAP Applications extractors
- SAP Applications ABAP
- SAP Applications BAPI (Business Application Programming Interface)
- SAP Applications IDoc
- SAP Master Data Services
- SAP NetWeaver Business Warehouse (BW)
- SAP NetWeaver BW Open Hub Destination service

The following table lists SAP interfaces, their intended functionality, and the types of data flows you can use with each.

Feature	How is this feature designed to be used?	Which data flow should be used?
SAP applications extractors	Use an SAP Applications datastore to import metadata from SAP extractors to create batch jobs that extract data from SAP application sources.	data flow or ABAP data flow
SAP Applications ABAP	An alternative to SAP extractors. Use an SAP Applications datastore to import metadata (tables, files, hierarchies, and functions) to create batch jobs to extract data from SAP application sources.	ABAP data flow
SAP Applications BAPI	Use an SAP Applications datastore to import BAPI function metadata to be included in batch or real-time jobs. BAPI functions can be used to update SAP application and SAP NetWeaver BW sources.	data flow
SAP Applications IDoc	Use an SAP Applications datastore to import IDoc metadata which can be used to create the following for batch and real-time jobs: For batch jobs • IDoc file sources • IDoc message sources • IDoc message targets For real-time jobs • IDoc file sources • IDoc message sources • IDoc file targets • IDoc message targets	data flow

Feature	How is this feature designed to be used?	Which data flow should be used?
SAP Master Data Services	Use an SAP Master Data Services datastore to import SAP Master Data Services metadata to create batch jobs that can be run from Data Services to read from SAP Master Data Services sources or update SAP Master Data Services targets.	data flow
SAP NetWeaver BW	Use an SAP BW Target datastore to import BW metadata to create batch jobs that can be run from SAP NetWeaver BW or Data Services to update SAP BW targets.	data flow (batch jobs only)
SAP NetWeaver BW Open Hub Destination service	Use an SAP BW Source datastore to import Open Hub Destination tables that are sources from which Data Services can read data.	data flow

1.3 Version checking

Data Services provides version information for SAP application ABAP programs and functions. With this feature, Data Services can:

- Embed version information in ABAP programs (programs that are both provided by Data Services for an SAP application environment and generated by Data Services from ABAP data flows)
- Generate warnings to update ABAP programs that were uploaded during a previous Data Services installation
- Compare an ABAP program with its ABAP data flow and produce a warning if it finds a version mismatch
- Compare functions on an SAP application server (uploaded during a previous Data Services installation) with those required by the current Data Services installation and produce a warning if it finds a version mismatch

To avoid version mismatches, it is recommended that when you upgrade Data Services, use it with the transport files supplied in the same product package and regenerate ABAP programs from ABAP data flows if you pre-loaded them to an SAP application server.

1.3.1 Generated ABAP programs

If you select **Execute Preloaded** for the **ABAP execution option mode** in your SAP Applications datastore, when Data Services executes a job that contains an ABAP data flow, it runs the previously generated and uploaded ABAP program. If this program was generated by an old version of Data Services, the job might not be taking advantage of the performance improvements in the latest version.

You can set Data Services to trigger version warning messages when a job executes. If you find a version mismatch, upgrade the generated ABAP program on the SAP server to match the Data Services version.

1.3.2 To upgrade a generated ABAP program stored on an SAP application server

1. Open a project containing the ABAP data flow that you want to update.
2. In the project area, select the ABAP data flow.
3. Select **Validation** > **Generate ABAP code**.
4. Modify your ABAP program on the SAP application server using this new code.

Note:
If you select the **Generate and Execute** mode for an SAP Applications datastore configuration, Data Services generates an ABAP program each time a job runs. Data Services does not need to check an ABAP program's version if the program is generated each time a job runs.

1.3.3 Data Services ABAP programs

When you connect to an SAP application environment, Data Services makes use of and tracks version information for built-in ABAP programs. If you encounter a warning, upgrade the Data Services ABAP programs to match the current Data Services version.

1.3.4 Data Services functions

If you want to connect to an SAP application environment, Data Services provides functions for you to load onto an SAP application server. Data Services tracks version information for these functions and can trigger warning messages if it finds that functions from a previous Data Services installation exist on the server. If you encounter a warning, upgrade the Data Services functions on the SAP application server to match the current Data Services version.

1.3.5 Warning messages

You can enable and disable version checking.

1.3.6 To enable or disable version checking for ABAP programs and functions

1. From Designer, select **Tools** > **Options** > **Job Server** > **General**.

   This dialog allows you to modify the configuration of the default Job Server.
2. Set the following keys in the al_engine section as needed.

   These keys control different SAP application warning messages:

Key	Messages
ABAP_PRELOAD_VERSION_MISMATCH	151201, 151203
ABAP_PRELOAD_KEY_MISMATCH	151202, 151204
ABAP_SUPPLIED_VERSION_MISMATCH	151205
SAP_FUNCTION_MISMATCH	151206

   The default for ABAP_PRELOAD_VERSION_MISMATCH and ABAP_PRELOAD_KEY_MISMATCH is FALSE, and the default value for ABAP_SUPPLIED_VERSION_MISMATCH and SAP_FUNCTION_MISMATCH is TRUE.

As described in the following sample scenarios, if you enable warnings for the ABAP and function version checking feature, Data Services provides an appropriate response:

Scenario	Response
After installing a new release of Data Services and enabling the version warning messages feature, you try to use Data Services against function metadata imported by an older Data Services version.	Data Services generates a warning message: 151206 3 3 4 "Warning: a mismatch was detected for SAP function <%s (%s)> : installed version <%s>. The expected version is <%s>. Please upload the correct function."
You just upgraded to a new version of Data Services but chose not to upgrade the SAP application server and you do not want to encounter warning messages.	You do nothing. However, it is recommended that you avoid this scenario by regenerating and uploading ABAP files when you upgrade Data Services.
You generate ABAP and upload it to your SAP server. After enabling all version warning messages and running it in execute_preloaded mode for a while, you change an ABAP data flow.	Data Services generates a warning message.
You generated an ABAP program and uploaded it to your SAP server. After enabling all version warning messages and running it in Execute Preloaded mode for a while, you upgrade Data Services.	Data Services generates a warning message because of the potential benefits you will gain by regenerating ABAP.

1.3.7 /BODS/GET_VERSION function

This function is included in each transport file for SAP applications.

/BODS/GET_VERSION retrieves the list of Data Services provided functions and their versions from the SAP application server.

I / P_FUNC / RFCFUNC-FUNCNAME

E / O_DI_VER / RFCFUNC-FUNCNAME

T / ENTRIES / TAB512

Installing Functions on the SAP Server

SAP BusinessObjects Data Services provides functions that support the use of the ABAP, BAPI, and IDoc interfaces on SAP servers. You will need some or all of these functions in the normal operation of the software in an SAP environment. These functions perform such operations as dynamically loading and executing ABAP programs from Data Services, efficiently running preloaded ABAP programs, allowing for seamless metadata browsing and importing from SAP servers, and reporting the status of running jobs. Some of these functions read data from SAP NetWeaver BW sources.

You must upload the provided functions to your SAP server in a production environment. It is recommended that you always upload the functions to your SAP server whether you are in a development, test, or production environment. The functions provide seamless integration between Data Services and SAP servers.

The default installation places two function module files for SAP servers in the `...\Data Services\Admin\R3_Functions\transport` directory. You then upload these files to SAP servers using the SAP Correction and Transport System (CTS) or manually. Using CTS allows for version control as the functions evolve across releases.

The installation provides two versions of transport files (depending on the server version you are using) to install the functions on the SAP server. To obtain the names of the latest transport files for installing or upgrading these SAP server functions, see the readme.txt file in the `...\Data Services\admin\R3_Functions\transport` directory.

2.1 Installing the functions using CTS

If you are installing SAP BusinessObjects Data Services for the first time or upgrading to a new release, you must add the provided functions to the SAP server. The recommended method is to use the SAP Change and Transport System (CTS). This procedure assumes you are familiar with CTS and have the authorization to perform the transport.

To install provided functions using the CTS system:

1. Copy the provided transport files to the appropriate directories on your SAP server.

   Note:
   In these steps, *`900XXX.XXX`* is a variable. To substitute the correct file name for the current release, see the readme.txt file in the `...\Data Services\admin\R3_Functions\transport` directory.

   a. Copy *`R900XXX.XXX`* to the `/usr/sap/trans/data` directory.
   b. Copy *`K900XXX.XXX`* to the `/usr/sap/trans/cofiles` directory.

2. Log on to the SAP server and run the transaction `/nSE37` to determine if function group /BODS/BODS already exists. The CTS system will install the Data Services functions into a single function group /BODS/BODS that it automatically creates on the SAP server if it does not already exist.

   If function group /BODS/BODS exists and contains previously installed Data Services functions, add an unconditional mode 2 option (parameter U2) to the `tp import` command described in step 4 below (`tp import <xxx>K900<xxx> <SID> U2`).
3. From a command window, run the following transport command:

   ```
   tp addtobuffer <xxx>K900<xxx> <SID>
   ```

   (where SID is the SAP system ID)

   You receive the response:

   ```
   This is tp version <SAP TP and SAP versions> for
       <database type> database. Addtobuffer successful for
       <xxx>K900<xxx> tp finished with a return code: 0 meaning:
       Everything OK
   ```
4. Run the next transport command:

   ```
   tp import <xxx>K900<xxx> <SID>
   ```

   (where SID is the SAP System ID)

   You receive a response that indicates the SAP TP and SAP versions, database type, SAP server name, and the TP return code, which indicates whether or not the TP command successfully uploaded the functions.
5. Run `/nSE37` again to verify that the functions uploaded.

When importing the functions, you might see an error such as the following:

```
1 of 1 requests do not match the component version of the target system
```

To prevent this error, in STMS enable the **Ignore Invalid Component Version** option.

2.2 Function modules

The names of the SAP BusinessObjects Data Services functions have the prefix /BODS/ added to the corresponding SAP function names. The appropriate Data Services function modules for your SAP version are installed when you upload them to your SAP server.

2.2.1 Packaging of /BODS/RFC_ABAP_INSTALL_AND_RUN

For SAP servers, SAP BusinessObjects Data Services provides a function called /BODS/RFC_ABAP_INSTALL_AND_RUN in its transport file. This function is used in the generate_and_execute mode (set in an SAP datastore) to generate ABAP code on Data Services and execute it on an SAP server. This function allows dynamic program generation and is particularly useful in a development environment. However, it also provides a level of write access to an SAP server that might not be acceptable to your security environment. If this is a concern, remove this function from your SAP server.

2.2.2 Sample function module

The following example illustrates the function module /BODS/SYNTAX_CHECK:

Example:

```
FUNCTION /BODS/SYNTAX_CHECK.
*"--------------------------------------------
*"*"Local interface:
*"    EXPORTING

    *"         VALUE(ERRORMESSAGE) LIKE SY-MSGV1
*"    TABLES
  *"         PROGRAM STRUCTURE PROGTAB OPTIONAL
*"--------------------------------------------
DATA: MESS(72), LIN(72), WRD(72).
   SYNTAX-CHECK FOR PROGRAM MESSAGE MESS LINE LIN WORD WRD.
  IF MESS <> SPACE.
  ERRORMESSAGE = MESS.
  EXIT.
   ENDIF.
ENDFUNCTION.
```

2.2.3 Function parameters

The following table describes information about functions, their interfaces, and function source code. This table describes the parameters with the following format:

```
Parameter type / Parameter name / Reference field
```

The parameter types are:

- I: Import parameter
- E: Export parameter
- T: Table parameter

Function name	Description

/BODS/ABAP_RUN	Executes generated or preloaded ABAP programs.
	`I / PROGRAMNAME / SY-REPID`
	`E / ERRORMESSAGE /  SY-MSGV1`
	`T / SELTAB /  RSPARAMS`
	`LOG /  LISTZEILE`
	Exception: PROGRAM_DOES_NOT_EXIST

/BODS/AUTH_IMPORT	Imports SAP application authorization profiles used by /BODS/ABAP_RUN.
	`I / AUTHPROF /  USR10-PROFN`
	`T / AUTHOBJ /  LISTZEILE`
	Exception: SECURITY_PROFILE_DOESNOT_EXIST

/BODS/BW_QUERY	Gets metadata information from an SAP BW system.
	`I / P_CODE / DD01D-DATATYPE`
	`I / P_IAREA / DD01D-DDTEXT`
	`I / P_ICUBE / DD01D-DDTEXT`
	`I / P_ODSO / DD01D-DDTEXT`
	`T/ ENTRIES / TAB512`

/BODS/COLUMN_SEARCH	/BODS/COLUMN_SEARCH is currently not used and is reserved for future implementation.
	`I / P_TNAME / DD03VT-DDTEXT`
	`I / P_CNAME / DD03VT-DDTEXT`
	`I / P_CDESC / DD03VT-DDTEXT`
	`I / P_LANG / DD03VT-DDLANGUAGE`
	`I / P_MAXROWS / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/DATA_PROFILE	Profiles an SAP table.
	` I / P_CODE / DD02L-TABNAME`
	` I / P_TABLE / DD02L-TABNAME`
	`I / P_FIELD / DD03L-FIELDNAME`
	`T / WRITES / LISTZEILE`

/BODS/EXTRACTOR_IMPORT	Allows import of an ERP extractor metadata into the Data Services repository.
	`I / P_LANG / DD02T-DDLANGUAGE DEFAULT SY-LANGU`
	`I / P_NAME / ROOSOURCE-OLTPSOURCE`
	`E / P_ERRORMESSAGE / SY-MSGV1`
	`T / P_RESULT / LISTZEILE`
	`T / P_VIEW_DEF / LISTZEILE`
	Exceptions:
	• EXTRACTOR_IS_NOT_SUPPORTED
	• NO_AUTHORITY
	• GET_METADATA_FAIL
	• EXTRACTOR_DOES_NOT_EXIST

Function name	Description

/BODS/EXTRACTOR_NAVIGATE	Allows various ways of listing available ERP extractors in an SAP system.
	`I / P_LANG / DD02T-DDLANGUAGE DEFAULT SY-LANGU`
	`I / P_APPLNM / ROOSOURCE-APPLNM`
	`I / P_TYPE / ROOSOURCE-TYPE`
	`I / P_ALLEXT / CHAR_01`
	`T / P_RETURN / LISTZEILE`

/BODS/EXTRACTOR_SEARCH	Allows searches for ERP extractors.
	`I / P_LANG / DD02T-DDLANGUAGE DEFAULT SY-LANGU`
	`I / P_NAME / ROOSOURCE-OLTPSOURCE`
	`I / P_MAX_ROWS / I`
	`I / P_FLAG / I`
	`T / P_RETURN / LISTZEILE`

/BODS/FILE_ROWCOUNT	Puts/gets the row count information for each transported file in the data flow into/from the /BODS/BODS table.
	`I / NAME / /BODS/BODS-NAME`
	`I / ROWCOUNT / /BODS/BODS-TOTAL_ROW`
	`I / SET / SONV-FLAG`
	`E / TOTAL_ROW / /BODS/BODS-TOTAL_ROW`

/BODS/FUNCTION_GET	Gets the function interface.
	`I / FUNCNAME / TFDIR-FUNCNAME`
	`T / PRMTAB / CATFU`

/BODS/GET_VERSION	Retrieves the list of Data Services provided functions and their versions from the SAP application server. This function is included in each transport file.
	`I / P_FUNC / RFCFUNC-FUNCNAME`
	`E / O_DI_VER / RFCFUNC-FUNCNAME`
	`T / ENTRIES / TAB512`

/BODS/IDOC_IMPORT	Imports IDocs from SAP applications.
	`I / IDOCTYP / EDISYN-DOCTYP`
	`I / LANG /  EDISEGT-LANGUA`
	`T / SEGMENTS / LISTZEILE`
	Exception: IDOC_TYPE_DOESNOT_EXIST

/BODS/IDOC_SEARCH	Searches existing IDocs in SAP applications.
	`I / P_NAME / EDIDOT-DOCTYP`
	`I / P_DESC /  EDIDOT-DESCRP`
	`I / P_LANG / EDIDOT-LANGUA`
	`I / P_MAXROWS / SY-TABIX`
	`I / P_FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

Function name	Description

/BODS/JOB_LOG	Provides error handling and retrieves the SAP application job log when a job is cancelled. This function is required for the execute_preloaded access method you can define for an SAP Applications datastore.
	`I / JOBCOUNT / TBTCO-JOBCOUNT`
	`I / JOBNAME / TBTCO-JOBNAME`
	`I / HOST / TBTCO-BTCSYSREAX`
	`T / LOG / BTCTLE`
	Exceptions:
	• JOB_NOT_FOUND
	• JOB_LOG_NOT_FOUND
	• JOB_NUMBER_NOT_VALID
	• LOGNAME_WRONG_FORMAT
	• LOGNAME_MISSINGLOG_HANDLE_ERROR
	• LOG_IS_EMPTY
	• LOG_NOT_FOUND
	• TEMSE_CONVERSION_NOT_POSSIBLE
	• TEMSE_ERROR
	• TEMSE_FUNCTION_ERROR

/BODS/JOB_RUN	Submits ABAP jobs for background execution.
	`I / PROGRAMNAME / SY-REPID`
	`I / JOBGROUP /  TBTCO-JOBGROUP`
	`I / JOBNAME / TBTCO-JOBNAME`
	`I / HOST / TBTCO-BTCSYSREAX`
	`I / JOBCLASS / TBTCO-JOBCLASS`
	`E / JOBCOUNT /  TBTCO-JOBCOUNT`
	`E / JOB_RELEASED / BTCH0000-CHAR1`
	`T / SELTAB /  RSPARAMS`
	Exceptions:
	• ABAP_PROGRAM_SYNTAX_ERROR
	• CANT_CREATE_JOB
	• INVALID_JOB_DATA
	• JOBNAME_MISSING
	• CANT_START_IMMEDIATE
	• INVALID_STARTDATE
	• JOB_CLOSE_FAILED
	• JOB_NOSTEPS
	• JOB_NOTEX
	• LOCK_FAILED
	• ABAP_PROGRAM_DOES_NOT_EXIST

Function name	Description

/BODS/JOB_STATUS	Queries ABAP job status after a job is successfully submitted.
	`I / JOBNAME / TBTCO-JOBNAME`
	`I / JOBCOUNT / TBTCO-JOBCOUNT`
	`I / HOST / TBTCO-BTCSYSREAX`
	`E / STATUS / TBTCO-STATUS`
	Exception: JOB_NOT_FOUND

/BODS/MODEL_NAVIGATE	Allows you to navigate through the SAP data model, helping to identify tables.
	`I / P_OID / DM41S-DMOID`
	`I / P_LANG / DD02T-DDLANGUAGE`
	`T / P_RETURN / LISTZEILE`

/BODS/READ_TEXT	Allows extraction of text from SAP applications. This function is an RFC-enabled version of the SAP application function module, READ_TEXT.
	`I / CLIENT / SY-MANDT`
	`I / ID / THEAD-TDID`
	`I / LANGUAGE / THEAD-TDSPRAS`
	`I / NAME / THEAD-TDNAME`
	`I / OBJECT / THEAD-TDOBJECT`
	`I / ARCHIVE_HANDLE / SY-TABIX`
	`E / HEADER / THEAD`
	`T / LINES / TLINE`
	Exceptions:
	• ID: Text ID invalid
	• LANGUAGE: Invalid language
	• NAME: Invalid text name
	• NOT_FOUND: Text not found
	• OBJECT: Invalid text object
	• REFERENCE_CHECK: Reference chain interrupted
	• WRONG_ACCESS_TO_ARCHIVE: Archive handle invalid

/BODS/RFC_ABAP_ INSTALL_AND_RUN	Allows Data Services to dynamically generate and run programs on an SAP server.
	`I / MODE / SY-MSGTY`
	`I / PROGRAMNAME / SY-REPID`
	`E / ERRORMESSAGE / SY-MSGV1`
	`T / PROGRAM / PROGTAB`
	`T / WRITES / LISTZEILE`

Function name	Description

/BODS/RFC_READ_EXTRACTOR	Allows extraction of data from an ERP extractor.
	`I / P_NAME / ROOSOURCE-OLTPSOURCE`
	`I / P_DELIMITER / SONV-FLAG`
	`I / P_ROWCOUNT / SOID-ACCNT`
	`I / P_IS_VIEWDATA / CHAR_01`
	`I / P_WHERE /  /BODS/CH2K-WA`
	`I / P_SIZE / RSIDOCSIZE`
	`E / P_OUT_TABLE / DD02L-TABNAME`
	`T / P_FIELDS / RFC_DB_FLD`
	`T / P_TBLOUT128 / /BODS/CH128`
	`T / P_TBLOUT512 / /BODS/CH512`
	`T / P_TBLOUT2048 / /BODS/CH2K`
	`T / P_TBLOUT8192 / /BODS/CH8192`
	`T / P_TBLOUT30000 / /BODS/CH30K`
	Exceptions:
	• DATA_BUFFER_EXCEEDED
	• EXTRACTOR_INIT_FAIL
	• EXTRACTOR_GETNEXT_FAIL
	• NO_AUTHORITY
	• GET_METADATA_FAIL
	• SYNTAX_ERROR
	• FIELD_NOT_VALID

/BODS/RFC_READ_TABLE	Allows extraction of SAP application table data.
	`I / QUERY_TABLE / DD02L-TABNAME`
	`I / DELIMITER / SONV-FLAG DEFAULT SPACE`
	`I / NO_DATA / SONV-FLAG DEFAULT SPACE`
	`I / ROWSKIPS / SOID-ACCNT DEFAULT 0`
	`I / ROWCOUNT / SOID-ACCNT DEFAULT 0`
	`T / OPTIONS / RFC_DB_OPT`
	`T / FIELDS / RFC_DB_FLD`
	`T / DATA / /BODS/CH2K`
	Exceptions:
	• TABLE_NOT_AVAILABLE
	• TABLE_WITHOUT_DATA
	• OPTION_NOT_VALID
	• FIELD_NOT_VALID
	• NOT_AUTHORIZED
	• DATA_BUFFER_EXCEEDED

Function name	Description

/BODS/RFC_READ_TABLE2	Allows extraction of SAP application table data (enhanced version of /BODS/RFC_READ_TABLE).
	`I / QUERY_TABLE / DD02L-TABNAME`
	`I / DELIMITER / SONV-FLAG`
	`I / NO_DATA / SONV-FLAG`
	`I / ROWSKIPS / SOID-ACCNT`
	`I / ROWCOUNT / SOID-ACCNT`
	`E / P_OUT_TABLE / DD02L-TABNAME`
	`T / OPTION S/ RFC_DB_OPT`
	`T / FIELDS / RFC_DB_FLD`
	`T / TBLOUT128 / /BODS/TAB128`
	`T / TBLOUT512 / /BODS/TAB512`
	`T / TBLOUT2048 / /BODS/TAB2048`
	`T / TBLOUT8192 / /BODS/TAB8192`
	`T / TBLOUT30000 / /BODS/TAB30K`
	Exceptions:
	• TABLE_NOT_AVAILABLE
	• TABLE_WITHOUT_DATA
	• OPTION_NOT_VALID
	• FIELD_NOT_VALID
	• NOT_AUTHORIZED
	• DATA_BUFFER_EXCEEDED

/BODS/RFC_READ_TABLE_FILE	Allows extraction of SAP application table data (writes the data to the specified file).
	`I / QUERY_TABLE / DD02L-TABNAME`
	`I / DELIMITER / SONV-STATUS`
	`I / NO_DATA / SONV-FLAG`
	`I / OUT_DIR / / BODS/CH2K`
	`I / DATA_FILE / /BODS/CH2K`
	`I / EMPTY_FILE / SONV-FLAG`
	`T / OPTIONS / RFC_DB_OPT`
	`T / FIELDS / RFC_DB_FLD`
	Exceptions:
	• TABLE_NOT_AVAILABLE
	• TABLE_WITHOUT_DATA
	• FIELD_NOT_VALID
	• NOT_AUTHORIZED
	• DATA_BUFFER_EXCEEDED

/BODS/SYNTAX_CHECK	Performs a syntax check for generated or preloaded ABAP. This function is required for the execute_preloaded access method you configure in an SAP Applications datastore.
	`E / ERRORMESSAGE / SY-MSGV1`
	`T / PROGRAM /  PROGTAB`

Function name	Description

/BODS/TABLE_IMPORT	Imports SAP application table definitions into the Data Services repository.
	`I / TABNAME / DD03L-TABNAME`
	`I / LANG / DD02T-DDLANGUAGE`
	`E / DESCRIPTION / DD02T-DDTEXT`
	`E / TABCLASS / DD02L-TABCLASS`
	`T / COLUMNS / LISTZEILE`
	`T / INDEXES / LISTZEILE`
	`T / FKEYS / LISTZEILE`
	Exceptions: NO_TABLECLASS_FOR_TABLE

/BODS/TABLE_SEARCH	Allows searches for SAP application tables.
	`I / P_NAME / DD02-DDTEXT`
	`I / P_DESC / DD02VV-DDTEXT`
	`I / P_LANG / DD02V-DDLANGUAGE`
	`I / P_MAXROWS / SY-TABIX`
	`I / P_FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/TEXTS	Allows extraction of texts for program areas (currently not used).
	`I /P_CODE/PROGTAB-LINE`
	`T /ENTRIES/TAB512`

/BODS/TREE_IMPORT	Allows you to import ERP hierarchy metadata into the Data Services repository.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`
	`E / DESCRIPTION / DD04T-DDTEXT`
	`E / CONTROLLING_AREA / DD02D-DDTEXT`
	`E / CHART_OF_ACCT / DD02D-DDTEXT`
	`E / TABLE_NAME / DD02D-TABNAME`
	`E / FIELD_NAME / DD03D-FIELDNAME`
	`E / DOMAIN_NAME / DD03L-DOMNAME`
	`E / CHECK_TABLE / DD03L-CHECKTABLE`
	`E / CONT_AREA_LIST / TAB512-WA`
	`T / COLUMNS / LISTZEILE`

/BODS/TREE_NAVIGATE	Allows GUI-based navigation of an SAP application environ-ment.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`

/BODS/TREE_PROF	Reads an ERP hierarchy for profiling.
	`I / P_GROUP / DD04L-ROLLNAME`
	`I / P_TABLE / DD02D-TABNAME`
	`T / WRITES / LISTZEILE`

Function name	Description

/BODS/TREE_SEARCH	Allows GUI-based search and selection in an SAP application environment.
	`I / TREENAME / DD08T-MESTEXT`
	`I / TREEDESC / DD04T-DDTEXT`
	`I / LANG / SY-LANGU`
	`I / MAX_ROWS / SY-TABIX`
	`I / FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/TREE_NAVIGATE40	Provides a GUI-based method to import tables and fields for SAP R/3 version 4.x and later environments.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`
	`T / P_RETURN / LISTZEILE`

/BODS/TREE_IMPORT40	Provides a GUI-based method to import tables and fields for SAP R/3 version 4.x and later environments.
	`I / P_LANG / DD04T-DDLANGUAGE`
	`I / P_TREEAREA / DD02D-DDTEXT`
	`I / P_TREEGROUP / DD04L-ROLLNAME`
	`I / P_TREETABLE / DD02D-TABNAME`
	`E / DESCRIPTION / DD04T-DDTEXT`
	`E / CONTROLLING_AREA / DD02D-DDTEXT`
	`E / CHART_OF_ACCT / DD02D-DDTEXT`
	`E / TABLE_NAME / DD02D-TABNAME`
	`E / FIELD_NAME / DD03D-FIELDNAME`
	`E / DOMAIN_NAME / DD03L-DOMNAME`
	`E / CHECK_TABLE / DD03L-CHECKTABLE`
	`E / CONT_AREA_LIST / TAB512-WA`
	`T / COLUMNS / LISTZEILE`

/BODS/TREE_SEARCH40	Allows GUI-based search and selection in an SAP R/3 version 4.x environment.
	`I / TREENAME / DD08T-MESTEXT`
	`I / TREEDESC / DD04T-DDTEXT`
	`I / LANG / SY-LANGU`
	`I / MAX_ROWS / SY-TABIX`
	`I / FLAG / SY-TABIX`
	`T / P_RETURN / LISTZEILE`

/BODS/UPLOAD	Uploads a Data Services-generated ABAP program to the SAP server.
	`I / P_PNAME / SY-REPID`
	`I / P_TITLE / RS38M-REPTI`
	`I / P_DEVC / TDEVC-DEVCLASS`
	`I / P_APPL / TRDIR-APPL`
	`I / P_RSTAT / TRDIR-RSTAT`
	`I / P_REQNUM / E070-TRKORR`
	`I / P_TSKNUM / E070-TRKORR`
	`T / PROGRAM / PROGTAB`
	`T / ENTRIES / TAB512`

Function name	Description

/BODS/RFC_STREAM_READ_TABLE	Allows data streaming via RFC protocol.
	`I / QUERY_TABLE / TABNAME`
	`I / CALLBACK_FM / FUNCNAME`
	`I / DELIMITER / SONV-FLAG`
	`I / BATCH_SIZE / INT4`
	`I / NUMB_OF_LINES / INT4`
	`T / PROJECTIONS / RFC_DB_OPT`
	`T / SELECTIONS / RFC_DB_OPT`

2.3 SAP applications security levels

You can choose from three security levels based on your system requirements. Typically, application development environments have four application phases: development, consolidation, test, and production. ABAP programs are generated in the development phase, consolidated from multiple developers, transported to the test phase for testing, and finally transported to the production phase. The Data Services security mechanisms were designed using the SAP application paradigm. The following table displays the various authorization levels and application phases.

User	Authorization level	Application phase
DEVUSER	High	Design or Design and Consolidation
TESTUSER	Middle	Test or Consolidation and Test
PRODUSER	Low	Production

A typical application passes through four phases:

1. In the design phase, a high-security profile user designs the data and work flows, and generates ABAP programs. The ABAP programs are dynamically communicated to SAP and executed.

   After execution, ABAP programs are automatically deleted.

   To do this, the developer must have authorization that could be designated DEVUSER.

2. In the consolidation phase, developers upload generated ABAP programs into their SAP system with predefined program names and preregistered correction numbers. A user in that phase can run these programs with a lower security profile, TESTUSER, for example.

   Consolidation can also occur in the design phase.

Here TESTUSER can create ABAP programs, install SAP functions, and schedule jobs.

3. In the test phase, the Data Services repository must be exported to a new repository, and the generated ABAP programs must be transported to the SAP test system. This is usually performed by TESTUSER.
4. The production phase begins when no additional program modifications are required.

   The user involved in the move to the production phase has the lowest security profile. The user in this phase could be called PRODUCER.

SAP authorization can be managed in two ways:

1. Profile generation method

   Every user is assigned to activity groups. Each activity group contains information about allowed activities (such as transaction and menu path). Only users with an activity authorization can perform that activity.

2. Profiles assigned to users method

   Every profile contains authorizations. The user administrator creates new authorizations, assigns them to profiles, and assigns profiles to users.

Data Services provides various authorization choices to meet your organization's needs:

- Some organizations use DEVUSER during all the application phases.
- Others do not want to assign DEVUSER authorization, which requires a high security profile, to any individual. In Data Services you can use the TESTUSER authorization to:
  - Generate ABAP on the development system
  - Upload it to SAP, then execute
- PRODUSER typically has the lowest authorization. This user can usually only execute ABAP programs to which TESTUSER has been granted authorization.

Data Services provides the checking mechanisms necessary to prevent unauthorized users from executing Data Services-generated programs in your SAP system. If you specify a security profile in SAP datastores (see Defining SAP Applications datastores), Data Services verifies authorization before executing the program.

If you do not specify a security profile, the Data Services programs do not perform a security check prior to execution.

2.4 SAP user authorizations

This section describes how to define a profile using the profile generation method and the specific user authorization level method. It also lists recommended SAP applications security profiles and authorizations for using Data Services.

2.4.1 Creating an SAP applications profile using profile generation

1. Create a user (for example for SAP DEV, TEST, or PRD systems).
2. Open the SAP Profile Generator (transaction PFCG).
3. Create an Activity group or Role.
4. Enter a description for the role.
5. Go to the **Authorizations** tab and click **Change authorization data**.
6. On the **Change Role: Authorizations** screen, click the **Manually**, toolbar icon.
7. The **Manual Selection of Authorizations** window opens.
8. Enter the required authorizations for this role.
9. Click **OK**
10. Return to the **Change Role: Authorizations** screen.
11. Manually configure components by entering the values documented in the section Authorizations for Data Services.
12. To complete the security profile, click the **Back** icon (or press F3), select the **User** tab, enter your SAP user ID for Data Services, and click the **Save** icon.

2.4.2 Defining a Data Services-specific authorization level

1. Determine which SAP profile you need.

   Required profiles depend on how you use Data Services. See SAP application profiles.
2. In SAP, create the required authorizations for your profiles.

   The required authorizations for your profiles are listed in SAP application profiles. Settings for each authorization are listed in Authorizations for Data Services.
3. Create an SAP profile.

   For example, you might create a profile specific to developing applications in Data Services: DS_DEV.
4. Assign the SAP authorizations to the SAP profile.
5. Assign the profile to an SAP user for Data Services.

2.4.3 SAP application profiles

How you use Data Services determines the profiles you need to define:

- Development and test profile
- Production profile

For information about the SAP BW loading profile, see SAP NetWeaver BW loading profile.

2.4.3.1 Development and test profile

To support development and test, create an SAP profile such as DS_DEV, that defines authorizations for DEVUSER or TESTUSER. This profile requires the following authorizations.

- S_BTCH_JOB
- S_DEVELOP
- S_RFC
- S_TABU_DIS
- S_TCODE

Related Topics

- Authorizations for Data Services

2.4.3.2 Production profile

To support production, create an SAP profile such as DS_PROD that defines authorizations for PRODUSER. This profile requires the following authorizations:

- S_BTCH_JOB
- S_RFC
- S_TABU_DIS
- S_TCODE

Related Topics

- Authorizations for Data Services

2.4.4 Development versus production functions

User permissions differ between development and production environments. A user with S_DEVELOP authorization can execute any function. If that authorization isn't present, the ABAP code checks the authorization object ZSDS.

Note that you can create custom authorization checks for any function by modifying the sample ABAP code in the shipped file sample_badi_impl.txt (admin\R3_Functions\ManualInstall).

2.4.4.1 Development-only functions

The following functions should be used only in development environments, not on production systems:

/BODS/AUTH_IMPORT

/BODS/EXTRACTOR_IMPORT

/BODS/FUNCTION_GET

/BODS/IDOC_IMPORT

/BODS/RFC_ABAP_INSTALL_AND_RUN

/BODS/TABLE_IMPORT

/BODS/TREE_IMPORT

/BODS/TREE_IMPORT40

/BODS/UPLOAD

2.4.4.2 Production functions

If the system is not a development system, then the user can only execute the following limited set of functions.

/BODS/BW_QUERY

/BODS/EXTRACTOR_NAVIGATE

/BODS/EXTRACTOR_SEARCH

/BODS/FILE_ROWCOUNT

/BODS/GET_VERSION

/BODS/IDOC_SEARCH

/BODS/JOB_LOG

/BODS/JOB_STATUS

/BODS/MODEL_NAVIGATE

/BODS/READ_TEXT

/BODS/SYNTAX_CHECK

/BODS/TABLE_SEARCH

/BODS/TEXTS

/BODS/TREE_NAVIGATE

/BODS/TREE_NAVIGATE40

/BODS/TREE_SEARCH

/BODS/TREE_SEARCH40

/BODS/COLUMN_SEARCH

/BODS/ABAP_RUN

/BODS/JOB_RUN

/BODS/RFC_READ_EXTRACTOR

/BODS/RFC_READ_TABLE

/BODS/RFC_READ_TABLE2

/BODS/RFC_READ_TABLE_FILE

/BODS/TREE_PROF

/BODS/DATA_PROFILE

/BODS/RFC_STREAM_READ_TABLE

2.4.5 Authorizations for Data Services

This section describes the authorizations that support Data Services operations.

For improved security, avoid using wildcards, generic, or blank values for authorization fields, especially in a production environment. Enter more specific values that are appropriate to your business applications.

2.4.5.1 Open Hub

The necessary authorizations to use the Open Hub interface in Data Services are contained in the S_BI-WHM_RFC profile.

In addition, Data Services needs the following authorization to work with the Open Hub interface:

S_RFC_ADM

Purpose: This object includes authorization checks for accessing individual administration functions in transaction SM59

Use: DEV, PROD

Class: Cross-application Authorization Objects

Text (Description): Administration for RFC Destination

Field	Values
Activity	03
RFCTYPE	T
RFCDEST	List of RFC destinations the user is allowed to access
ICF_VALUE	Authorizations for destination in transaction SM59

2.4.5.2 G_800S_GSE

Purpose: This authorization allows Data Services to access ERP hierarchies.

Use: DEV, PROD

Text (Description): Special Purpose Ledger Sets: Set

Class: Financial Accounting

Field	Values
Authorization group	Not used
Activity	03

2.4.5.3 S_BTCH_JOB

Purpose: This authorization checks batch job release privileges.

Use: DEV, PROD

Text (Description): Batch processing

Class: Basis

Field	Values
Job operation	RELE
Summary of jobs for a group	Not used

2.4.5.4 S_CTS_ADMI

Purpose: This authorization allows Data Services to perform CTS operations.

Use: DEV

Text (Description): Administration Functions in Change and Transport System

Class: Basis: Administration

Field	Values
Administration Tasks for Change and Transport System	PROJ

2.4.5.5 S_DEVELOP

Data Services uses the S_DEVELOP authorization in several ways.

Purpose: This implementation allows Data Services to perform a column search.

Use: DEV, PROD

Text (Description): ABAP Workbench

Class: Basis - Development Environment

Field	Values
Package	List of packages for tables that a user is allowed to access
Object type	TABL
Object name	List of tables that a user is allowed to access
Authorization group ABAP/4 program	Not used
Activity	03

Purpose: This authorization allows Data Services to run generated programs on the SAP server.

Use: DEV

Text (Description): ABAP Workbench

Class: Basis - Development Environment

Field	Values
Package	$TMP
Object type	PROG
Object name	List of temporary program names that are allowed to be generated
Authorization group ABAP/4 program	Not used
Activity	01 and 02

Purpose: This implementation allows Data Services to import a table or to search for a table.

Use: DEV, PROD (table search)

Text (Description): ABAP Workbench

Class: Basis - Development Environment

Field	Values
Package	List of packages for tables that a user is allowed to access

Field	Values
Object type	VIEW, TABL and TTYP
Object name	List of tables and views that a user is allowed to access
Authorization group ABAP/4 program	Not used
Activity	03

2.4.5.6 S_IDOCDEFT

Purpose: This authorization allows Data Services to work with IDocs.

Use: DEV, PROD

Text (Description): WFEDI: S_IDOCDEFT - Access to IDoc Development

Class: Basis - Central Functions

Field	Values
Activity	03
Extension	Not used
Basic type	Not used
Transaction Code	WE30

2.4.5.7 S_RFC

Purpose: This authorization allows users to execute remote functions on an SAP server.

Use: DEV, PROD

Text (Description): Authorization check for RFC access

Class: Cross-application authorization object

Field	Values
Activity	16
Name of RFC to be protected	BAPI, CADR, RFC1, SCAT, SDIF, SLST, SUNI, SUTL, SDTX, SYST, /BODS/BODS, RSAB, SDIFRUNTIME, and any other required function group
Type of RFC object to be protected	FUGR

2.4.5.8 S_RS_ADMWB

Purpose: Use this authorization for BW loading.

Use: DEV, PROD

Text (Description): Administrator Workbench - Objects

Class: Business Warehouse

Field	Value
Administrator Workbench object	WORKBENCH, SOURCESYS, APPLCOMP, IN-FOAREA, INFOOBJECT, INFOPACKAG, ODSOBJECT
Activity	03

Related Topics
• SAP NetWeaver BW as a target

2.4.5.9 S_RS_ICUBE

Purpose: This authorization allows Data Services to access an InfoCube.

Use: DEV, PROD

Class: Business Information Warehouse

Text (Description): Data Warehousing Workbench - InfoCube

Field	Values
InfoArea	List of InfoAreas that a user is allowed to access
InfoCube	List of InfoCubes that a user is allowed to access
InfoCube Subobject	DEFINITION
Activity	03

2.4.5.10 S_RS_ODSO

Purpose: This authorization allows Data Services to access a DataStore Object.

Use: DEV, PROD

Text (Description): Data Warehousing Workbench - DataStore Object

Class: Business Information Warehouse

Field	Values
InfoArea	List of InfoAreas that a user is allowed to access
DataStore Object	List of DataStore Objects that a user is allowed to access
Subobject for ODS Object	DEFINITION
Activity	03

2.4.5.11 S_SCRP_TXT

Purpose: This authorization allows Data Services to read SAP texts.

Use: DEV, PROD

Text (Description): SAPscript: Standard text

Class: SBOP Data Services Authorization Object

Field	Values
Language Key	List of language keys that a user is allowed to access
Text ID	List of text IDs that a user is allowed to access
Name	List of text names that a user is allowed to access
Activity	SHOW

2.4.5.12 S_TABU_DIS

Purpose: This authorization allows Data Services to access table data in an SAP system.

Use: DEV, PROD

Text (Description): Table Maintenance (via standard tools such as SM30)

Class: Basis

Field	Value
Activity	03
Authorization group	Table groups that a user is allowed to access

2.4.5.13 S_TCODE

Data Services uses the S_TCODE authorization in several ways.

Purpose: This authorization grants the user access to specific transactions.

Text (Description): Authorization check for transaction start

Class: Cross-application authorization object

Field	Value
Transaction Code	SE37, SE38, SU53

Purpose: This authorization allows Data Services to execute functions in the Data Warehousing Workbench.

Use: DEV, PROD

Text (Description): Transaction Code Check at Transaction Start

Class: Cross-application Authorization Objects

Field	Values
Transaction Code	RSA1

In addition, you should have access to the contents of the following tables:

RSDAREA

RSDAREAT

RSDCUBE

RSDCUBET

RSDODSO

RSDODSOT

2.4.5.14 S_TRANSPRT

Purpose: This authorization allows Data Services to access the Transport Organizer.

Use: DEV

Text (Description): Transport Organizer

Class: Basis - Development Environment

Field	Values
Request Type (Change and Transport System)	DTRA
Activity	01

2.4.5.15 S_USER_GRP

Purpose: This authorization allows Data Services to establish a connection to the SAP server.

Use: DEV, PROD

Text (Description): User Master Maintenance: User Groups

Class: Basis: Administration

Field	Values
User group in user master maintenance	User group for Data Services user

2.4.5.16 S_USER_PRO

Purpose: This authorization allows Data Services to import an authorization profile.

Use: DEV

Text (Description): User Master Maintenance: Authorization Profile

Class: Basis: Administration

Field	Values
Auth. profile in user master maintenance	Authorization Profile to be imported
Activity	03

2.4.5.17 ZDSAUTH

Purpose: This authorization gives a user an access to Data Services functions.

Use: DEV, PROD

Text (Description): SBOP Data Services - general authorization

Class: SBOP Data Services Authorization Object

Field	Values
ACTVT: Activity	16 (Execute)

2.4.5.18 ZDSDEV

Purpose: This is the general authorization object (Data Services-specific version of S_DEVELOP).

Use: DEV, PROD

Text (Description): SBOP Data Services Authorization Object for development

Class: SBOP Data Services Authorization Object

Field	Values
Package	List of packages for tables that a user is allowed to access
Object type	VIEW, TABL, and TTYP
Object name	DD objects that a user is allowed to access
Authorization group ABAP/4 program	Not used
Activity	03

2.4.5.19 ZPGMCHK

Purpose: This authorization determines which programs may execute in a production environment.

Use: PROD

Text (Description): SBOP Data Services Authorization Object for program names

Class: SBOP Data Services Authorization Object

Field	Values
ACTVT: Activity	16 (Execute)
PROGRAM: ABAP program name	Program names that are allowed to be executed in a production environment

2.4.5.20 ZSDS

Purpose: This authorization lets you to define whether the SAP system should be treated as a development or production system from the perspective of Data Services.

Use: DEV, PROD

Text (Description): Data Services Authorization Object for functions

Class: SBOP Data Services Authorization Object

Field	Values
ACTVT	Not used
ZSYSTYPE	D: Development system Any other value: Production system

2.4.5.21 Browsing metadata for an SAP BW source datastore

In addition, if you need to be able to browse metadata for an SAP BW source datastore, you should have access to the contents of the following tables:

RSDAREA

RSDAREAT

RSDCUBE

RSDCUBET

RSDODSO

RSDODSOT

Enabling security between SAP and Data Services

3.1 Authentication using Secure Network Communications (SNC)

Enabling SNC provides a secure connection between SAP systems and Data Services.

This procedure describes how to configure Authentication using SNC between Data Services clients for secure communications from SAP systems.

For more information about SNC security, on the SAP Service Marketplace see the *SNC User's Guide* at https://service.sap.com/~form/sapnet?_SHORTKEY=00200797470000074724.

1. In Data Services, verify installation of a 64-bit SNC library.
2. Complete the procedure "Configuring the Use of the SAP Cryptographic Library for SNC" including creating a trusted certificate. This document is available on the SAP Service Marketplace Web site at https://service.sap.com/~form/sapnet?_SHORTKEY=00200797470000074724.
3. Add or open the datastore editor to configure and click **Advanced** to display the **Authentication** options. For **Authentication**, select **SNC**. Complete the remaining SNC options as applicable.

Related Topics

- Datastore
- Secure Network Communications (SNC) in BW

3.2 Security between SAP BW and the Data Services RFC server

Data Services provides two separate options for enabling security between SAP BW and the Data Services RFC server.

- Access Control List (ACL): ACL security authorizes user access permissions to Data Services repositories through the SAP BusinessObjects Central Management Server.
- Secure Network Communications (SNC): SNC provides a secure connection between SAP BW and the Data Services RFC server.

Sample usage scenarios:

- In a private network where the SAP user's identity must be verified to access a Data Services repository, implementing the ACL option is sufficient.

- In a public network but with trusted SAP users, implemeting the SNC option is sufficient.
- In a public network where the SAP user's identity must be verified to access a Data Services repository, implement both the ACL and SNC options.

For more information about SAP security, on the SAP Service Marketplace Web site see https://service.sap.com/security.

3.2.1 Access Control List (ACL)

Enabling ACL security authorizes and authenticates users for access to Data Services repositories from SAP BW using the SAP BusinessObjects Central Management Server (CMS). The interface for the CMS is the Central Management Console (CMC). For help using the CMC, after logging in click the question mark icon.

This procedure also includes configuration steps in the SAP BW system and in the Data ServicesAdministrator.

1. In the CMC, authenticate the SAP BW system as follows:
   a. Select the **Authentication** management area from the drop-down list.
   b. Double-click the **SAP** object to open it.
   c. On the **Entitlement Systems** tab, select the system to authenticate and click **Update**.
   d. On the **Role Import** tab, add the SAP role(s) to import (that correspond to the Data Services repositories you want to secure) and click **Update**.
   e. On the **User Update** tab under "Update Roles and Aliases", click **Update Now** to import the users associated with the imported SAP roles.

      Note:
      You can also select **Schedule** to schedule regular updates to the user list.

   f. Close the dialog box.

   To ensure the SAP roles and users have been imported, select the CMC **Users and Groups** management area. Select the **User List** and **Group List** (or **Group Hierarchy**) nodes to view the imported users and roles. “Roles” are referred to as “groups” in the CMC.

2. In the CMC, set group and user rights for access to Data Services repositories as follows:
   a. Select the **Data Services** management area from the drop-down list.
   b. Open (double-click) the repository to configure.
   c. Select **User Security**.
   d. Click **Add Principals**.
   e. Select the user or group to configure and click **Add and Assign Security**.
   f. On the **Access Levels** tab, assign the desired access level(s) and click **Apply.**
   g. To add or remove rights for principals, click the **Advanced** tab > **Add/Remove Rights**. You can grant only the following general global rights for Data Services:

      Delete objects: Delete objects only

      Edit objects: Edit, delete, or view objects

View objects: View objects only

h. To grant a principal the right to execute batch jobs in Data Services, select the **Applications** managemant area and open the **Data Services Application** object. Repeat steps 2c through 2g, except on the **Add/Remove Rights** dialog box, you can grant the specific right **Execute batch job**.

3. In the SAP BW system, configure the profile parameters login/create_sso2_ticket=2 and login/accept_sso2_ticket=2. You can use transaction rz11. See the **Maintain profile parameters** > **Profile parameter maintenance** interface.

   For more information, refer to the procedure "Changing Profile Parameters for SSO" on the SAP Help Portal at http://help.sap.com/saphelp_nw70/helpdata/en/45/590250736a6628e10000000a1553f6/frameset.htm.

4. In the SAP BW system, configure RFC destination authentication as follows:
   a. Use transaction sm59 to access RFC Destinations.
   b. Expand **TCP/IP connections**.
   c. Open the RFC destination to configure.
   d. On the **Logon and Security** tab, select **Send Assertion Ticket**.

      The assertion ticket contains user authentication information.
   e. Save the configuration.

5. In the Data ServicesAdministrator, enable ACL security as follows:
   a. Configure the RFC server if you haven't already.

      Navigate to **SAP Connections** > **RFC Server Interface** > **RFC Server Interface Configuration** > **Add**. For more information, see the Data Services *Management Console Guide*.
   b. On the RFC Server Configuration page, expand the **Server security** node and select the **Enable ACL check** box.

Related Topics

• Secure Network Communications (SNC) in BW

3.2.2 Secure Network Communications (SNC) in BW

Enabling SNC provides a secure connection between SAP BW and the Data Services RFC server.

This procedure describes how to configure SNC between SAP BW and the Data Services RFC server for jobs that are launched from SAP BW.

1. In Data Services, verify installation of a 64-bit SNC library.
2. Complete the procedure "Configuring the Use of the SAP Cryptographic Library for SNC" including creating a trusted certificate. This document is available on the SAP Service Marketplace Web site at https://websmp207.sap-ag.de/~form/sapnet?_SHORTKEY=00200797470000074724.

3. Taking the results from the procedure in step 2, navigate to the Data Services Administrator "RFC Server Configuration" page to enable SNC security (**SAP Connections** > **RFC Server Interface** > **RFC Server Interface Configuration** and select the server name):
   a. Expand the **Server security** node.
   b. Select **Enable SNC**.
   c. Enter the **Server SNC name**.
   d. Enter the **SNC library path**.
   e. Select the desired **SNC quality of protection** from the drop-down list.

Related Topics

- Access Control List (ACL)
- Authentication using Secure Network Communications (SNC)

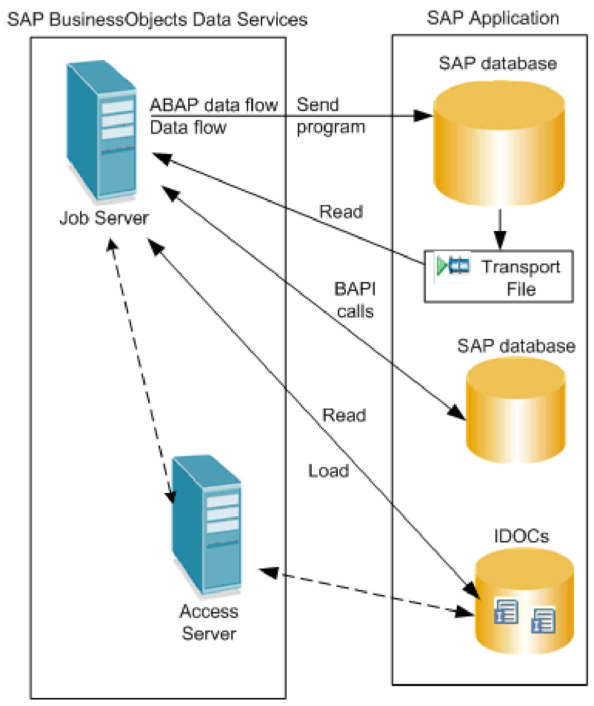
Connecting to SAP Applications

Data Services provides several methods for moving data into and out of SAP applications:

- Streaming data from SAP using RFC: Reads data from SAP applications using regular data flows and supports tables and extractors.
- ABAP programming language: Reads data from SAP applications using ABAP data flows and supports tables, hierarchies, extractors, and functions with scalar arguments.
- RFC/BAPI: Loads data to SAP applications and allows direct access to application tables outside of the data flows through RFC function calls. Because a function is handled as a nested column within a query, it can return SAP tables, structures, or scalar parameters, which can be loaded into any target.

  Data Services can process any RFC-enabled function call including all available parameters plus user-supplied functions. These functions are often used to write certain information into an SAP system or read data that has to be calculated by the function.
- IDoc interface: Reads from and loads data to SAP applications. Data Services can send, receive, and create SAP IDocs including extended and custom IDocs and supports reduced input sets, parallel processing, real-time, and batch processing.

The following diagram shows a high-level overview of the components involved with reading data from and loading data to SAP applications.



Note: Access Server is used for listening for IDocs in real-time jobs

You can search an SAP application server for tables, IDocs, functions, hierarchies, and extractors. You import metadata using an SAP Applications datastore connection.

Once you have imported metadata, Data Services provides:

- ABAP data flows: Move table and hierarchy data between SAP applications and Data Services using ABAP.
- Data flows in batch jobs: Can contain ABAP data flows (as sources), tables, hierarchies, and extractors (as sources), or IDocs (as sources or targets), and SAP RFC-enabled functions.
- Data flows in real-time jobs: Can contain SAP tables, IDocs as sources and targets, and RFC-enabled functions.

4.1 SAP Applications datastores

This section discusses the steps required to use an SAP application as a Data Services data source:

- Defining SAP Applications datastores
- Browsing, searching, and importing metadata from SAP applications

4.1.1 Defining SAP Applications datastores

An SAP application datastore includes the following information:

- Connection information including the application server name, the language used by the SAP client application, the client and system numbers, and optionally reference to an sapnwrfc.ini file.

  You supply these values based on your installation of the SAP application.
- Data transfer method used to exchange information between Data Services and the SAP application.

  Select the method and supply the appropriate information based on decisions you make about running Data Services jobs.
- Security information, specifically the SAP security profile to be used by all connections instigated from this datastore between Data Services and the SAP application. You supply a security profile based on your installation of the SAP application.

Related Topics
- Data transfer methods
- SAP applications security levels

4.1.1.1 To define an SAP Applications datastore

1. In the object library, select the **Datastores** tab.
2. Right-click inside the object library window and choose **New**.

   The datastore editor opens.
3. Enter a unique name for the datastore in the **Datastore name** box.

   The name can contain alphanumeric characters and underscores. It cannot contain spaces.
4. For **Datastore type**, select **SAP Applications**.
5. Enter the **Application server** name.

   Refer to your SAP documentation for the appropriate syntax.
6. Enter the **User name** and **Password** information.
7. You can either save the datastore or add more information to it:
   - To save the datastore and close the Datastore Editor, click **OK**.
   - To add more parameters, click **Advanced**.

     To enter values for each configuration option, click the cell next to each configuration name. For details, see the *SAP BusinessObjects Data Services Reference Guide*.

Related Topics

- Datastore

4.1.1.2 Using the sapnwrfc.ini file

To change or add parameters without modifying the datastore settings, you can specify them in an sapnwrfc.ini file.

Place your sapnwrfc.ini file in the same directory as the RFC client/server program (for Data Services, commonly %LINK_DIR\bin). Or you can specify the DS_NWRFC_INI environment variable in the operating system. For example, you could can place the sapnwrfc.ini file in C:\Temp and specify `DS_NWRFC_INI=C:\Temp`.

You can specify an sapnwrfc.ini file for two different applications in Data Services:

- To use an sapnwrfc.ini file to override datastore settings, edit the SAP datastore in the Designer. From the object library, open the editor for the SAP datastore to configure and click **Advanced**. Under the "SAP" option category, for **Use sapnwrfc.ini**, select **Yes** and enter the appropriate **Destination name**.
- To use an sapnwrfc.ini file for real-time IDocs, configure the RFC client in the Administrator. Expand the **Real-Time** node, and for the Access Server to configure, click **Client Interfaces**. Open an instance of the RFC client, and select the check box to **Use sapnwrfc.ini** and enter a destination name.

Note:
Data Services ignores the parameters USER, PASSWD, and CLIENT in sapnwrfc.ini because storing these security-related parameters could introduce a security risk.

For more information about how to use the sapnwrfc.ini file, refer to the SAP documentation at http://help.sap.com/saphelp_nwpi711/helpdata/en/48/ce50e418d3424be10000000a421937/frameset.htm.

4.1.1.3 SAP security profiles

You can associate a security profile with a datastore so that data flows that access SAP application sources defined by the datastore include appropriate authorization checking.

SAP security profiles are useful for:

- Creating profiles of users that can access particular ABAP programs
- Moving ABAP programs from development to test and production

Specify a security profile in the Advanced options in the datastore editor. By default, Data Services does not use any SAP security profile.

Specify any security profile defined in SAP (a predefined profile or a profile you defined). Authorization checks are embedded in generated ABAP code.

Related Topics

• SAP user authorizations

4.1.2 Browsing, searching, and importing metadata from SAP applications

The processes for browsing, searching, and importing metadata from SAP applications are similar to the processes you would use for any source system.

There are several types of SAP objects for which you can import metadata into the Data Services object library. You can import metadata for each one by name. You can import metadata for hierarchies, IDocs, and tables by searching. You can also browse SAP applications to find hierarchies and tables.

SAP item	Methods for importing metadata
Functions	Name
Hierarchies	Name, search, browse
IDocs	Name, search
Tables	Name, search, browse
Extractors	Name, search, browse

4.1.2.1 Importing metadata by name

1. In the object library, select the **Datastores** tab.
2. Right-click the SAP Applications datastore you want to use and select **Import by Name**.

   The Import by Name window opens.
3. In the **Type** box, select the SAP item to import.
4. In the **Name** box, enter the name SAP uses for the item you want to import. For example, enter SISC001 for an IDoc.

   Note:
   Hierarchies require entering a controlling area and set table.
5. Click **OK**.

4.1.2.2 Importing table and hierarchy metadata by browsing

Tables and hierarchies can be imported by browsing the data available through the SAP datastore.

For more information about importing SAP application hierarchies, see Extracting data from SAP application hierarchies.

To import table and hierarchy metadata by browsing:

1. In the object library, select the **Datastores** tab.
2. Right-click the datastore you want to use and choose **Open**.

   The items already imported from the datastore appear in the workspace.
3. Click **External Data**.

   Click the plus sign next to a folder icon to navigate the structure of tables or hierarchies. Tables appear with a table icon.

   Hierarchies are also nested according to their use in SAP.
4. Select the items for which you want to import metadata.

   For example, to import a table, you must select a table rather than a folder that contains tables. To import a hierarchy, select the hierarchy group that contains the hierarchy.
5. Right-click and choose **Import**.
6. In the object library, expand the datastore to display the list of imported objects.

4.1.2.3 Importing IDoc metadata using the Search option

To use IDocs, you must first import the metadata from SAP into your repository. IDocs can be imported by name or by using the Search option.

1. In the object library, click the **Datastores** tab.
2. Right-click an SAP Applications datastore and select **Search**.

   The "Search" window opens.
3. Verify the datastore or repository in which to search displays in the **Look in** box, select **External data**, and for "Object type" select **IDocs**.
4. Enter search text in either the **Name** or **Description** box:
   - In the **Name** box enter the name or part of the name of the IDoc you want to find.
   - In the **Description** box, enter a full or partial description instead of the name.
5. Click **Search**.

A list of objects appears in the window.

6. Right-click the name of the IDoc you want to import and select **Import**.

   The IDoc appears in the object library under its datastore and under the IDoc category.

   Data Services imports the schema for the IDoc type, maintaining hierarchical relationships among the data fields. Double-click the IDoc type in the object library to display the IDoc schema.

4.1.2.3.1 Adding an IDoc to a job

1. Select an imported IDoc from the object library.
2. Drag the IDoc into a data flow.
3. Select an appropriate option from the pop-up menu: **Make IDoc file source**, **Make IDoc file target**, **Mark IDoc message source**, or **Make IDoc message target**.

4.2 Reading from SAP extractors

This section describes how to browse, import, and read from SAP extractor sources.

Data Services can read from extractors in the following ways:

- ODP data sources: Refers to extractors that use the Operational Data Provider data replication API (ODP data replication API). In Data Services, you can browse these extractors' hierarchy of applications. Data Services supports all types supported by the ODP data replication API in addition to changed-data capture (CDC) functionality (referred to as "delta" in SAP applications).
- Native: Refers to extractors that do not use the ODP data replication API. Data Services supports the TRAN and ATTR types only, and changed-data capture is not available.

Related Topics

• Extractor source

4.2.1 Browsing extractors

Use an SAP Applications datastore to connect to the extractor sources. Then in the object library on the Datastores tab, right-click the datastore and click **Open** (or double-click the datastore) to view the datastore explorer.

With **External Metadata** selected, expand the Extractors node to display the available extractors organized in following subfolders:

- "All": Displays an alphabetical list of all extractors
- "By Application": Displays the extractors hierarchically by application and type

Related Topics

- To define an SAP Applications datastore

4.2.2 Importing extractor metadata

You can import extractors by name, by browsing, or by searching.

To import by browsing, in the datastore explorer with **External Metadata** selected, right-click an extractor and click **Import**.

In the dialog box, enter a unique combination for **Name of consumer** and **Name of project** to identify the subscription for this extractor.

- **Name of consumer**: Use a system name such as "Data Services."
- **Name of project**: For a given consumer (such as "Data Services"), you have the option of creating several independent subscriptions. For example, you might want to access the same DataSource in full or changed-data capture (delta) mode with different selection criteria. Therefore, enter a unique project name here to identify this subscription.

For example, for the **Name of consumer** "Data Services" and **Name of project** "Extraction for Data Warehouse 1," you might create initial (full) and CDC data flows for a customer dimension table. Both data flows would use this same instance (subscription) of the extractor. But for a second project that loads another data warehouse, you could import the extractor a second time using the same consumer name but a different project name such as "Extraction for Data Warehouse 2" to make the two subscriptions independent.

For the "Extraction mode" options, if the extractor is not an ODP data source, the only option is to import the extractor in **Query** mode. These extractors do not support changed-data capture. In Query mode, the extractor returns all data and pushes down WHERE clauses.

If the extractor is an ODP data source, **Changed-data capture (CDC)** is the default extraction mode. Here you can optionally configure changed-data capture (delta) parameters as described in the next section.

To import multiple extractors, Ctrl-click the extractors, right-click, and click **Import**. You can choose between **Changed-data capture** or **Query** mode for this set of extractors. In this case, configuration details for changed-data capture (CDC) are not available, but you can configure them for each extractor after importing.

After importing the extractor(s), you can also view properties such as the extractor type by right-clicking the object in the object library and clicking **Properties**. To view column properties, open the extractor object, right-click a column name, and select **Properties**.

Related Topics

- Browsing, searching, and importing metadata from SAP applications

4.2.3 Capturing changed data

Data Services supports changed-data capture (CDC) for extractors enabled with the Operational Data Provider (ODP) data replication API. You can configure different CDC options in several places:

- When importing or reimporting the extractor metadata from the datastore explorer
- On the extractor object properties **Extraction process** tab
- After importing the object into a data flow, in the object's source properties

4.2.3.1 Importing for CDC

When you import an ODP data source, the "Importing (extractor name) extractor" dialog box displays. Define the subscription by entering a **Name of consumer** and a **Name of project**.

To optionally configure CDC for the extractor during import:

1. Verify the Extraction mode **Changed-data capture (CDC)** is selected.

   The table for defining CDC filters is enabled.
2. For each filter, select a "Field name" using the drop-down box in the first cell.
3. The "Sign "and "Comparison" fields work together. Select a "Sign" to include or exclude a comparison operator. The comparison operators include:
   - Equals
   - Contains pattern
   - Between
4. Enter value(s) in the "From" and (optionally) "To" fields.

   Values are constant literals and cannot be variables or expressions. Use single quotes around strings.
5. Repeat the "Field name", "Sign", "Comparison", and From/To selections for each filter.
6. Click **Import.**

The object imports with a name of the format *`ExtractorName_NameofProject`*.

When reimporting an extractor (right-click the object name and select **Reimport**), you can also change the CDC filters at that time (however at that time the name fields cannot be changed).

Related Topics

• Importing extractor metadata

4.2.3.2 Editing CDC filters

After importing the extractor, you can also edit the CDC filters in the object properties.

1. In the object library, right-click the extractor name and select **Properties**.
2. Select the **Consumer details** tab.

   The "Name of consumer" and "Name of project" fields cannot be changed.
3. Edit the "Field name", "Sign", "Comparison", "From", and "To" values for each filter.
4. Click **OK**.

Note:
If you change the filter properties for an extractor, "reset" the job by running it with the source option **Initial load** selected. Then you can run the job with CDC filters applied (with **Initial load** cleared).

4.2.3.3 CDC columns

The import operation adds the following columns to the original table:

Column name	Description	Source
DI_SEQUENCE_NUMBER	Contains an integer for sequencing.	Data Services
DI_OPERATION_TYPE	Contains the row operation type: I (insert), D (delete), X (unknown)	Data Services
ODQ_CHANGEMODE	For CDC, identifies the row type. Works in conjunction with ODQ_ENTITYCNTR to determine how to handle the row. C (create), U (update), D (delete)	SAP
ODQ_ENTITYCNTR	For CDC, an integer that works in conjunction with ODQ_CHANGEMODE to determine how to the handle row.	SAP

The following table describes the relationships between ODQ_CHANGEMODE and ODQ_ENTITYCNTR.

CHANGEMODE	ENTITYCNTR	Description
C	+1	The record is new and inserted.
U	+1	The record is inserted. If there is a record with same key already available, the record is updated. In this case, there should be a corresponding U- (short notation for U and -1) record. The record contains the new information.
U	-1	The record is deleted and replaced with the corresponding record with same key and U+ (short notation for U and +1) if available. The record contains the previous (old/outdated) information.
U	0	The record is updated, but the key figures must be added to the already available key figures. So the record returns the difference between the old record and the new record, not the new record itself.
D	-1	The record is deleted. It contains the same key as an already existing record, but the key figures are inverted. So the record might be "deleted" by adding the key figures to the already available values. They should sum up to 0.
D	0	The record is deleted. It contains the actual data in both key and key figures.

4.3 Reading from SAP tables

You can use a regular data flow to process large volumes of data from SAP tables.

To improve the performance while reading large volumes of data, the source table editor includes the following options:

- The **Array fetch size** option allows the data to be sent in chunks, which avoids large caches on the source side.
- The **Execute in background (batch)** option lets you run the SAP table reader in batch mode (using a background work process) for time-consuming transactions.

Related Topics

- Source
- Batch execution mode
- Open Hub Table source

4.4 ABAP data flow processing

In Data Services, ABAP data flows extract SAP data from tables and follow this general execution process:

- In a development environment, Data Services generates the ABAP code, or in production, the program is uploaded to the SAP server
- Data Services connects to the SAP application server through remote function call (RFC).
- The SAP application server executes the ABAP code.
- The ABAP program results are generated and communicated back to Data Services, which then loads the target data cache.

Note:
Some operations such as uploading generated ABAP programs or using the Direct Download data transfer method require interoperability with the SAP GUI. If you experience connection problems with these operations, implement the appropriate Support Package as described in SAP Note 1258724, "Starting SAPGUI using external RFC libraries."

4.4.1 ABAP interface objects and options

The following Designer objects and options pertain to SAP.

4.4.1.1 Data flow objects

The Data Flows tab of the object library includes both the **Data Flows** and **ABAP Data Flows**.

You can place ABAP data flows into batch jobs as a source. ABAP data flows generate ABAP code. In most cases, you use a data transport object as the target of an ABAP data flow. A data transport specifies a staging file for data extracted from SAP applications.

ABAP data flows and data transport objects also appear in the tool palette.

See SAP-specific objects for details about these objects.

4.4.1.2 Data flow commands

With a data flow in the workspace, the following commands appear in the **Validation** menu:

- **Validate** > **Validate ABAP**

  Validates the ABAP code Data Services produces for the data flow
- **Generate ABAP**

  Displays the ABAP generated for the current data flow

4.4.1.3 SAP environment options

In the Designer, the following options are available under **Tools** > **Options** > **SAP** > **Environment**.

4.4.1.3.1 Maximum number of rows buffered before transfer to Data Services

The maximum number of rows held in the SAP buffer prior to writing to the data transport file. The default value is 5000 rows. This option lets you manage the amount of memory used by generated ABAP programs by selecting the number of rows to write in a given batch.

You set this value once and it applies to all SAP application connections initiated by Data Services.

4.4.1.3.2 Prefix for ABAP program names

You can define this parameter to separate Data Services-generated ABAP program names from other ABAP program names. Because ABAP program names must begin with a Y or a Z, this option provides a default that begins with ZW. The maximum for this option is two characters.

4.4.1.3.3 Convert SAP null to null

Converts NULL values from SAP sources in data flows to database-specific NULL. See Working with null values from SAP applications for more information.

4.4.1.3.4 Disable background SAP job status in Data Services log

Disables trace log entries when Data Services queries the status of jobs submitted in background (using the datastore option **Execute in background (batch)**).

4.4.1.4 Data transfer requirements

When you configure SAP Applications datastores in Data Services, you specify the data transfer method. This method defines how data that is extracted by the ABAP running on the SAP application server becomes available to the Data Services server (the computer on which the Data Services Job Server runs). The options are:

- RFC: Use to stream data from the source SAP system directly to the Data Services data flow process using RFC.
- Direct download: The SAP server transfers the data directly to the **Local directory** using the SAP-provided function GUI_DOWNLOAD or WS_DOWNLOAD.
- Shared directory: Default method. The SAP server loads the transport file into the **Working directory on SAP server**. The file is read using the **Application path to the shared directory** from the Job Server computer.
- FTP: The SAP server loads the **Working directory on SAP server** with the transport file. Then the Job Server calls an FTP program and connects to the SAP server to download the file to the **Local directory**.
- Custom Transfer: SAP server loads the **Working directory on SAP server** with the transport file. The file is read by a third-party file transfer (custom transfer) program and loaded to the **Custom transfer local directory**.

See Data transfer methods for more information.

4.4.1.5 SAP applications permissions

Data Services uses different execution methods to execute generated ABAP in design, development, test, and production environments. Each method has different security requirements. For detailed information and planning worksheets, see SAP applications security levels.

4.4.1.6 Security profile

ABAP programs generated by Data Services can verify the permissions of the user executing the program. If a security profile is provided, the generated ABAP verifies that the user executing the program has the permissions specified by that profile. This security profile should be transported to the production system prior to running Data Services against it.

4.4.2 Data flows

SAP applications commonly consist of three key components:

- SAPGUI client: A front-end application that provides a logical view of the entire SAP system

- Application layer: An application server that interprets ABAP, SAP's proprietary fourth-generation language, and generates and executes SQL statements against the database
- Database layer: A database manager that manages the SAP data in its tables

Interacting directly with the SAP application database layer can be problematic because:

- It is complex.
- The data contained in pool and cluster tables can be accessed only through ABAP, not through SQL.

The Data Services SAP ABAP interface allows you to extract SAP application data, through the application layer, using ABAP. This interface also provides the ABAP data flow and its associated Data ServicesDesigner options. This means that:

- You do not need to understand many of the complexities of the SAP database layer.
- You can use Data Services to access not only transparent tables, but also pool and cluster tables.
- You can call corresponding SAP functions to ensure the results match what SAP would display.

To use an ABAP data flow, create a batch job, create a data flow inside the job, then create an ABAP data flow inside the first data flow. This type of batch job uses the following general execution process:

- Data Services generates the ABAP code.
- Data Services connects to the SAP application server via remote function call (RFC).
- The ABAP code executes on the SAP application server.
- The ABAP program results are generated and communicated back to Data Services, which then loads the target data cache.

Related Topics

- Designer Guide: Naming data flows

4.4.2.1 What is an ABAP data flow?

An ABAP data flow extracts and transforms data from SAP application tables, files, and hierarchies. The ABAP data flow produces a data set that you can use as input to other transforms, save to a file that resides on an SAP application server, or save to an SAP table. When Data Services executes ABAP data flows, it translates the extraction requirements into ABAP programs and passes them to SAP to execute.

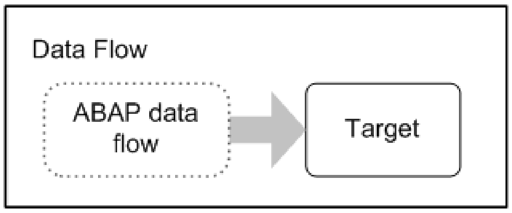
When you plan to use the SAP extracted data outside the ABAP data flow, the ABAP data flow can place the extracted data set in a file. A Data Services data transport passes the data in the file to the parent data flow. A data transport indicates how data is communicated between the SAP application server and Data Services.

When the target inside the ABAP data flow is the final destination for the data, you do not need a data transport. The ABAP data flow stores the data in a file saved on the SAP application server. In this case, the ABAP data flow must be the only object in a data flow.

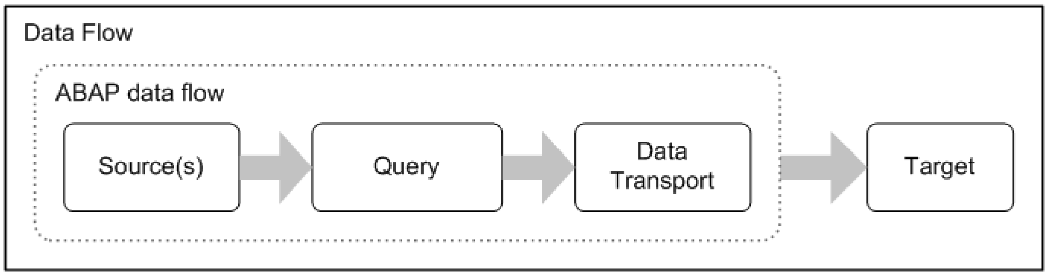
You can also use a data flow to extract data from an SAP table. When you use a data flow, Data Services extracts data from SAP using an RFC (remote function call). However, when you use an ABAP data flow, Data Services extracts data using ABAP and performance is generally better.

Because SAP performs all operations in an ABAP data flow, you want the operations in an ABAP data flow to make best use of SAP system capabilities. Therefore, the preferred method of use involves ABAP.

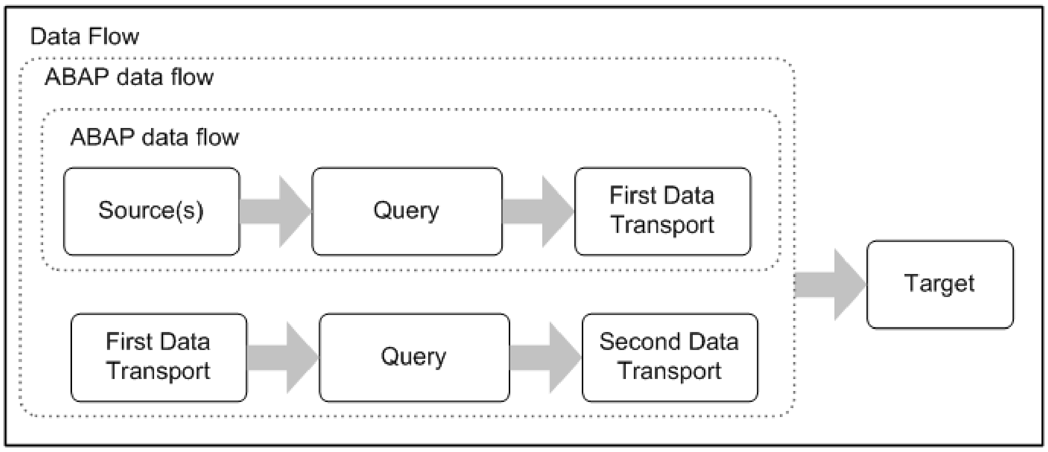
Consider the simple case of extracting data from an SAP application table and loading it into a non-SAP target. The parent data flow contains two objects: an ABAP data flow and a target.



The ABAP data flow defines the specific source tables from which you want to extract data and contains a data transport for the resulting data set.



In some cases, the transformations you need to perform in the SAP application are too complex to include in just one ABAP data flow. For example, the transformation might require multiple passes over the data. Data Services allows you to nest an ABAP data flow inside another ABAP data flow so that you can simplify each data flow. You can keep the output from the inner data flow in a file on the ABAP server and use the file in the outer data flow.



Note:
Although ABAP data flows are used in data flows, data flows cannot be placed in ABAP data flows. As shown above, only other ABAP data flows can be placed inside ABAP data flows. Also, ABAP data flows cannot be used as embedded data flows.

4.4.2.2 Defining ABAP data flows

This section describes the four procedures to complete when defining an ABAP data flow:

- Adding an ABAP data flow.
- Specifying sources
- Defining a query
- Specifying a target

4.4.2.2.1 Adding an ABAP data flow

This procedure describes how to first add and configure the data flow that will contain the ABAP data flow.

1. In the workspace, add a new data flow and open it.
2. On the tool palette, click the icon for an ABAP data flow.
3. Click a location in the workspace.

   The "Properties" window appears, prompting for the ABAP data flow options.
4. In the **Datastore** list, select the SAP datastore to use for the extraction.
5. In the **Generated ABAP file name** box, enter the file name for the ABAP program that this data flow generates.

   Data Services writes this file in the generated ABAP directory specified for the datastore.

6. Enter other options in the window.
   * **ABAP program name**

     Specifies the name Data Services uses for the ABAP program name that runs in the SAP application. It must begin with Y or Z.
   * **Job name**

     Specifies the name used for the job that runs in the SAP application. It defaults to the name of the data flow.
   * **ABAP row limit**

     Can be used for the test phase of development to reduce the number of data rows sent to the transport file. The default value of zero indicates that all data rows will be output to the transport file. When this value is set to a number greater than zero, only the specified number of rows go to the transport file.

     If you enter a number here, you can then enable/disable this feature from the **Tools** > **Options** control panel. The Designer and Job Server must be on the same computer. ABAP row limit is enabled/disabled globally for all SAP jobs associated with this Job Server. Remember to disable this feature before moving the jobs into a production environment.

     To enable or disable the option, select **Tools** > **Options** > **Job Server** > **General**. Enter data (case sensitive) as follows:

Field	Entry
Section:	AL_Engine
Key:	ABAP_ROW_LIMIT
Value:	TRUE

     Then click **OK**.
   * **Join rank**

     If the calling data flow uses output from this ABAP data flow (in the form of a data transport) in a join, join rank indicates the rank of the source relative to other sources in the data flow. The software joins sources with higher join ranks before joining sources with lower join ranks.

     Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify a join rank only in the Query transform editor.

     Join rank must be a non-negative integer. Default value is 0.
   * **Parallel process threads**

     Specifies the number of threads that read or load data. This option allows you to parallel process reading and loading.
   * **RFC data transfer package size**

With the **Data transfer method** of **RFC** selected at the datastore level, you can adjust the number of rows to process per data package to maximize streaming performance. Defaults to 5000 rows.

- **Cache**

  If the calling data flow uses output of the ABAP data flow as the inner source for a join, you can cache the output to improve the performance of the join. Use caching only if you are sure the data is small enough to fit in the memory allotted to the cache.

  Default setting is No.

  The option is ignored if the ABAP data flow is nested inside another ABAP data flow.

7. Click **OK**.
   An ABAP data flow appears in the workspace and the project area.
8. With the ABAP data flow name highlighted, type a new name.
9. Click the ABAP data flow name.
   Data Services opens the workspace where you next add and define the components of the ABAP data flow.

4.4.2.2.2 Specifying sources

You can add SAP application tables or hierarchies to an ABAP data flow as sources. This example adds a table.

Do not use non-SAP or IDoc sources here; add those sources to the parent data flow and use a query (including BAPI functions as needed) to join those sources with the data set produced by the ABAP data flow.

To specify data sources:

1. From an SAP Applications datastore, click an SAP table and drag it into the workspace of an open ABAP data flow.
2. Click the name of the source table in the workspace to open the source table editor.
   In the source table editor, you can set three options:
   - **Join rank**

     Indicates the rank of the source relative to other tables joined in the data flow. The software joins sources with higher join ranks before joining sources with lower join ranks.

     Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor.

     Must be a non-negative integer. Default value is 0.
   - **Cache**

     Indicates whether Data Services should read the required data from the table and load it into memory.

     Best practice is to specify the cache in the Query transform editor. When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source. In the Query editor, cache is set to Automatic by default. The automatic setting carries forward the setting from the source table.

There are three options in sources:

Cache option	Description
Yes	The table is always cached unless it is the outer-most table in a join.
No	The table is never cached. Package size operation is bypassed.
Automatic	Data Services determines whether or not to cache the table based on table size and number of filtered records.

The default option is No.

- **Package size**

  The package size option limits the number of rows brought into memory when building internal tables. By limiting the number of rows, less memory is used to construct internal tables.

  You should use this option only for cached tables whose size could lead to excessive memory consumption.

Related Topics
- Extracting hierarchy set data
- Optimizing inner joins
- Manually setting join order
- Tables used in a join
- Source

4.4.2.2.3 Defining a query

Define a query to produce the desired data set. With a query, you can manipulate SAP application sources as follows:

- Join the data sets from more than one source
- Select the rows to extract from the source
- Define the schema of the extracted data set
- Perform calculations on column values

A single query can have several inner joins or several left outer joins, but may not contain mixed inner joins and left outer joins.

Best practice when joining multiple sources is to set join rank and cache for all sources in the Query editor.

When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source. In the Query editor, cache is set to Automatic by default. The automatic setting carries forward the setting from the source table. The following table shows the relationship between cache settings in the source, Query editor, and whether the software will load the data in the source table into cache.

Cache Setting in Source	Cache Setting in Query Editor	Effective Cache Setting in ABAP data flow
Yes	Automatic	Yes
No	Automatic	No
Automatic (Data Services determines whether or not to cache the table based on table size and number of filtered records)	Automatic	Automatic (based on table size and number of filtered records)
Yes	Yes	Yes
No	Yes	Yes
Automatic	Yes	Yes
Yes	No	No
No	No	No
Automatic	No	No

Note:
If any one input schema has a cache specified in the Query editor, the software considers only Query editor cache settings and ignores all source editor cache settings.

Data Services supports using the results of an ABAP data flow inside another ABAP data flow. If you are using the data in another ABAP data flow, make sure to:

- Keep the data on the SAP application server rather than on the Data Services server.
- Use **FTP** or **Shared directory** as the data transfer method.

You specify the location of the output file and data transfer method in the SAP datastore associated with the ABAP data flow.

To define a query:

1. Add a query to the ABAP data flow.
2. Draw the data flow connections.
3. Use the query editor to specify the desired data set.

Related Topics

- Reference Guide: Transforms, Platform Transforms, Query

4.4.2.2.4 Specifying a target

Specify a target for the resulting data set. Potential targets include a data transport, which makes the data set available to the calling data flow, or a file on the SAP application server.

To define a data transport:

1. Add a data transport to the ABAP data flow and rename it.
2. Connect the components of the ABAP data flow and map the query.
3. Click the name of the data transport.

   The data transport editor appears in the workspace.
4. In the data transport editor, enter the **File name** of the transport file.

   The file is placed in the **Working directory on SAP server** specified in the datastore definition.
5. If necessary, you can specify that the output file will be appended to an existing file with the same name (presumably created in a previous run of this same job). The default behavior is to replace any existing file.
6. Click the back arrow to return to the ABAP data flow.

   Upon execution, the ABAP data flow populates the file designated in the **File name** box.

4.4.2.3 Extracting data from SAP application hierarchies

The Data Services hierarchy object can extract set hierarchy information from the following set types:

- Basic sets: Sets containing values from one dimension. For example, in a basic set such as Assets that uses the dimension Accounts, all values in Assets are account numbers or ranges of account numbers.
- Single-dimension sets: Combinations of basic sets and/or other single-dimension sets that contain values from the same dimension. For example, the basic set Assets can be combined with the basic set Liabilities because they both use the same dimension Accounts.

In particular, Cost Center and Cost Element hierarchies in the Controlling module of SAP ERP are supported set hierarchies.

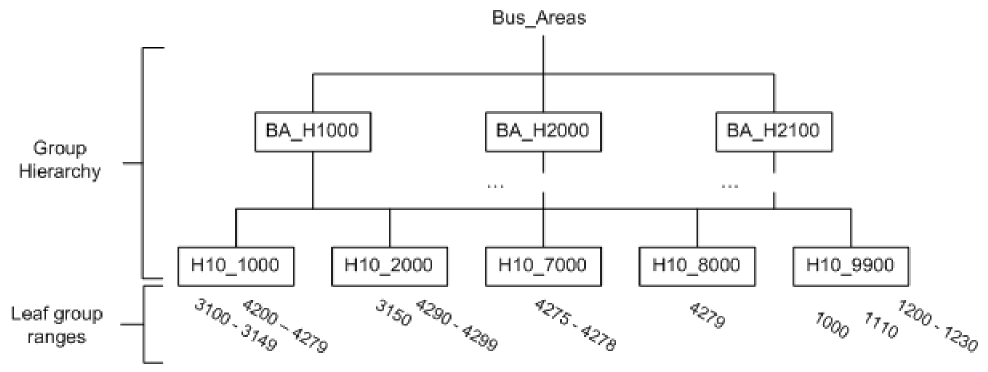
The following sections describe how hierarchies are defined in SAP applications and how Data Services translates the hierarchical relationships into horizontal or vertical representations:

- SAP application hierarchy sets
- Extracting hierarchy set data
- Example data flow with an SAP source

4.4.2.3.1 SAP application hierarchy sets

Hierarchy set information from SAP applications includes hierarchy nodes and ranges of values associated with the leaf nodes. The possible values are defined by the domain upon which the hierarchy is based. The values themselves are stored in the set table associated with the hierarchy.

The set data represented in this hierarchy is as follows:



4.4.2.3.2 Extracting hierarchy set data

Data Services provides a hierarchy object that extracts group nodes and leaf group ranges from supported SAP set hierarchies. When Data Services extracts the hierarchy using a hierarchy object, the result includes a row for each parent-child relationship in the hierarchy and associated leaf group ranges.

A hierarchy object produces the following output when extracting the Business area hierarchy from Controlling area 1000 shown previously.

	Parent_ID	Child_ID	VALUE_FROM	VALUE_TO
NULL parent to root node		0HBUS_AREAS		
Root node to first child	0HBUS_AREAS	0HBA_H1000		
First child (now parent) to its first child	0HBA_H1000	0HH10_1000	0000003100	0000003149
	0HBA_H1000	0HH10_1000	0000004200	0000004279
First child (now parent to its second child	0HBA_H1000	0HH10_2000	0000003150	0000003150
	0HBA_H1000	0HH10_2000	0000004290	0000004299
First child (now parent) to its third child	0HBA_H1000	0HH10_7000	0000004275	0000004278
First child (now parent) to its fourth child	0HBA_H1000	0HH10_8000	0000004279	0000004279

	Parent_ID	Child_ID	VALUE_FROM	VALUE_TO
First child (now parent) to its fifth child	0HBA_H1000	0HH10_9900	0000001000	0000001000
	0HBA_H1000	0HH10_9900	0000001110	0000001110
	0HBA_H1000	0HH10_9900	0000001200	0000001230
Root node to second child	0HBUS_AREAS	0HBA_H2000		
Second child (now parent) to its first child	0HBA_H2000	0HH20_3000	2-4200	2-4200
	0HBA_H2000	0HH20_3000	2-4210	2-4210
Root node to third child	0HBUS_AREAS	0HBA_H2100		
Third child (now parent) to its first child	0HBA_H2100	0HH21_9000	21-3100	21-3100
			…	

When you view the SAP hierarchy types in Data Services, you do not see the data associated with the hierarchy; instead, you see all of the hierarchy types grouped in two ways:

- Together in a single section (all hierarchy types organized across controlling areas by extraction logic)
- By controlling area

The names of the hierarchy types are made up of the SAP application set table name and the hierarchy group name. The hierarchies are named after the root node.

To extract data using a hierarchy, import the hierarchy metadata as part of an SAP application datastore definition.

To import a hierarchy type from an SAP application to your repository:

1. In the object library, select the **Datastore** tab and open the datastore corresponding to the SAP instance from which you want to import the hierarchy type.
2. In the workspace select **External Data**.
3. Expand the Hierarchy node.
4. Select the hierarchy type to import.

   The available hierarchies are listed according to the controlling areas with which they are associated. You can also browse through all hierarchies regardless of controlling area in the section labeled "All Controlling Areas."

5. Right-click and choose **Import**.

The internal view of the datastore shows the hierarchy type.

The imported hierarchy

After a hierarchy type is imported into a datastore, it appears in the object library.

A hierarchy instance in an ABAP data flow

Use a hierarchy object as you would a table from the object library: select the hierarchy, then drag it into the definition of the ABAP data flow open in the workspace.

The instance of the hierarchy object in the data flow has options you can set to specify:

- A single controlling area or all controlling areas from which the hierarchy data is extracted.

  If you imported the hierarchy type from a controlling area in the browser, or if you specified a controlling area when importing the hierarchy by name, a single controlling area is already specified in the hierarchy definition. If you imported the hierarchy type from the All Hierarchies browser section or did not specify a controlling area, you can specify the controlling area for each instance of the hierarchy.

  Choose a single controlling area from the list for best performance.
- A single hierarchy or all hierarchies in the hierarchy type.

  Choose a single hierarchy for best performance. Enter the hierarchy name as it is shown in the external browser of the datastore.

4.4.2.3.3 Example data flow with an SAP source

This section describes one technique for extracting and transforming the hierarchy so you can easily report on any level of the hierarchy.

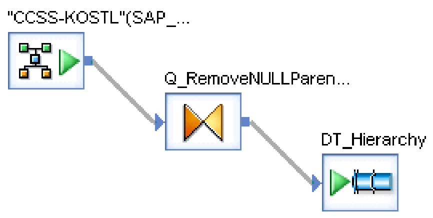
The job in this example consists of three data flows:

- Hierarchy_Extraction: an ABAP data flow that extracts the Business Areas hierarchy for Controlling Area 1000.
- Hierarchy_Leaf_Values: an ABAP data flow that associates the leaf data with the child nodes of the hierarchy.
- DF_Hierarchy_Flattening: a data flow that horizontally flattens the hierarchy.

Hierarchy_Extraction data flow

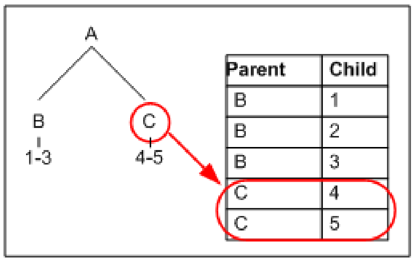
The Hierarchy_Extraction data flow is an ABAP data flow that extracts the hierarchy from SAP. It performs these operations:

1. Extracts the hierarchy type from SAP.
2. Filters NULL parent rows from the output.
3. Maps the parent, child, leaf and root flags, and controlling area columns to the output.



Hierarchy_Leaf_Values data flow

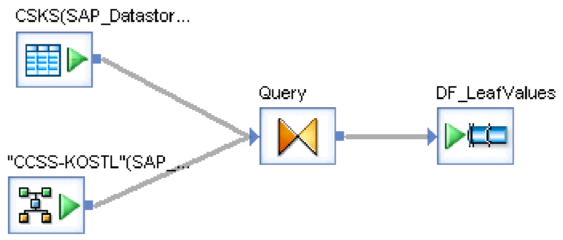
The Hierarchy_Leaf_Values data flow is an ABAP data flow that extracts values from the set table that correspond to the leaf nodes in the hierarchy. It produces a new set of parent/child pairs where the new parents are the leaf nodes from the hierarchy and the new children are the values associated with the leaf nodes. The result of this data flow is essentially another level to the hierarchy.



Parent	Child
B	1
B	2
B	3
C	4
C	5

Specifically, the data flow does the following:

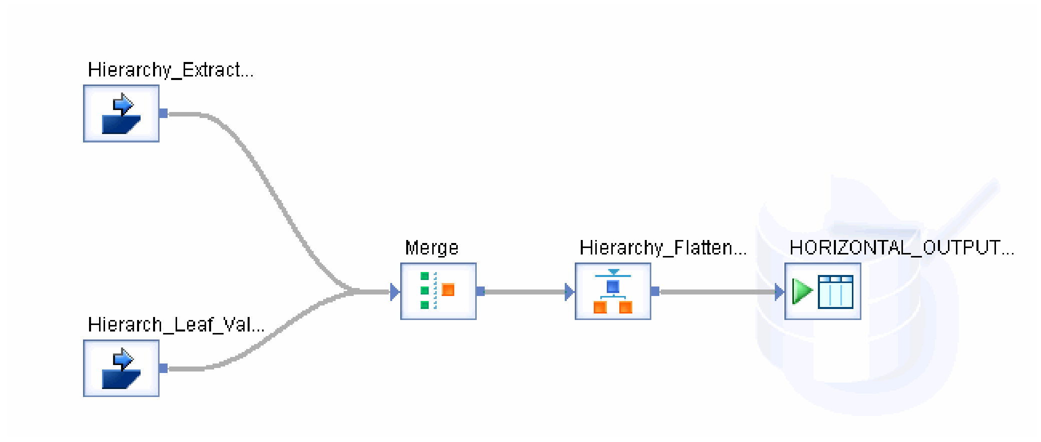
1. Joins the hierarchy with the set table to produce a row for each leaf group value, filtering for valid cost centers and Controlling Area 1000.
2. Maps the child_id and child_desc from the hierarchy to the parent_id and parent_desc in the output. Maps the cost center id (KOKRS) from the set table to the child_id in the output.
3. Looks up a text description of the cost center id and maps the result to the child_desc.



DF_Hierarchy_Flattening data flow

The DF_Hierarchy_Flattening data flow merges the leaf group values with the hierarchy and horizontally flattens the hierarchy as follows:

1. Merges the output from the two ABAP data flows.
2. Horizontally flattens the data using the Hierarchy_Flattening transform.
3. Loads the data into a target.



The output contains a row for each node in the hierarchy including each leaf group value. The table below shows a section of the output including the BA_H2000 cost center group:

Current leaf		Level 0	Level 1	Level 2	Level 3
0HBUS_AREAS	0	0HBUS_AREAS	0HBA_H2000		
0HBA_H2000	1	0HBUS_AREAS	0HBA_H2000		
0HH20_3000	2	0HBUS_AREAS	0HBA_H2000	0HH20_3000	
2-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	2-4200
2-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	2-4210
21-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-3100
21-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-3110
21-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4200
21-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4210
21-4280	3	0HBUS_AREAS	0HBA_H2000	0HH20_3000	21-4280
0HH20_4000	2	0HBUS_AREAS	0HBA_H2000	0HH20_4000	
2-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3100
2-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3110
2-3200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-3200
2-4100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4100
2-4120	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4120

Current leaf		Level 0	Level 1	Level 2	Level 3
2-4130	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4130
2-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4200
2-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	2-4210
21-3100	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-3100
21-3110	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-3110
21-4200	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4200
21-4210	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4210
21-4280	3	0HBUS_AREAS	0HBA_H2000	0HH20_4000	21-4280
0HH20_9900	2	0HBUS_AREAS	0HBA_H2000	0HH20_9900	
2-1110	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1110
2-1200	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1200
2-1210	3	0HBUS_AREAS	0HBA_H2000	0HH20_9900	2-1210
...		...	...	...	...

4.4.2.4 Creating custom ABAP transforms

You can create ABAP programs and incorporate them into an ABAP data flow as custom ABAP transforms. This section gives you information about creating and using custom transforms based on ABAP programs.

4.4.2.4.1 ABAP program requirements

You can map a custom-written ABAP FORM logic block to a Data Services custom ABAP transform. The ABAP logic block needs to perform the following functions:

- Extract data out of the SAP application using custom logic that is not currently supported by Data Services ABAP generation logic—for example, import statements and submit statements.

  Note:
  A subset of call function statements is supported: Functions returning scalar values are supported; ones returning tables are not.

- Load the resulting data into an output schema defined for the custom ABAP transform. You can also define and pass parameters to a custom ABAP transform. To include executing the ABAP logic block as part of your Data Services data movement, define a custom ABAP transform via the transforms tab. Then add it to an existing ABAP data flow.

Modifying custom ABAP for Data Services

Use an SAP FORM to modify a custom ABAP program. The custom ABAP must be enclosed in a FORM block to work with Data Services. In the structure shown below, notice the beginning of the FORM section after the comments. Use a FORM to pass the parameters to the output.

```
Comments
Start "FORM" block
Customer's custom ABAP
Start "OTAB1" block
End "OTAB1" block
End "FORM" block
```

Custom ABAP program tips

FORMNAME should be entered as it is shown at the beginning of the custom ABAP program. It will be replaced when the final ABAP program is generated.

$CONT_AREA and $ROOTNODE are parameters. Their data types are referenced in the ABAP. These optional parameters can be passed to the transform if defined in the Variables and parameters interface of the Designer and initialized.

After the FORM block, add a OTAB1 logic block immediately after the custom ABAP and before the ENDFORM line. The custom ABAP extracts the results from the SAP application server. The OTAB1 logic block takes these results and moves them into an internal table called OTAB1. You should not declare the OTAB1 in the FORM because it will be replaced in the final generated ABAP. The internal table OTAB1 is replaced by ITAB1 and made accessible to Data Services. Use the OTAB1 logic block as a guide, but it must be modified to follow the logic of your custom ABAP.

4.4.2.4.2 Using ABAP logic blocks in transforms

If you have ABAP logic blocks you want to use in your Data Services data movement, you can make them available as transforms.

To import an ABAP logic block

1. Open the object library to the **Transforms** tab.
2. Right-click the **Platform** node and choose **New ABAP Transform**.

   A new ABAP transform appears in the object library.
3. Right-click the transform and choose **Properties**.
4. Change the name of the transform so users can find it in the object library.
5. Click **OK**.
6. Right-click the transform and choose **Open** to define the custom ABAP transform.
7. For **ABAP language file name**, enter the full path name of the file containing the ABAP logic block.

You can also click the button next to the box to navigate to the file.

8. For **Join rank**, you can enter a weight to indicate the rank of this data set in a join. You can also set this value when the transform is used in an ABAP data flow.

To define the output schema created by the transform

1. In the output schema, right-click and choose **New Output Column**.

   A Column Properties window opens.

2. Enter the **Name** of the column.
3. Select the **Data type** of the column.
4. Enter the **Content type** of the column.
5. Enter a **Description** of the column.
6. Click **OK**.
7. Repeat for each column in the schema produced by the ABAP logic block.

To define the input and output parameters for the transform

1. From the **Tools** menu, choose the **Variables** option to open the Variables and Parameters window.

   The transform is the current context in the window.

2. Right-click in the window and choose **Insert**.
3. Right-click the new parameter and choose **Properties**.
4. Change the name of the parameter.

   Choose a **Name** that identifies this parameter so users can map appropriate data to it.

5. Set the **Data type** of the parameter.
6. Set the **Parameter type**.
7. Repeat to define additional parameters.

To use the transform in an ABAP data flow

1. Open an ABAP data flow in the workspace.
2. Drag the transform from the object library into the workspace.

   The transform appears in the ABAP data flow as a source.

3. Construct the rest of the ABAP data flow using the data schema produced by the transform.

To pass values to the transform

1. With the ABAP data flow open in the workspace, click **Tools** > **Variables**.

   The Variables and Parameters window opens with the ABAP data flow as its context.

2. Click the **Calls** tab.

3. To set a value for one or more parameter(s), double-click each parameter and specify an expression or value in the **Value** field.

4.4.2.4.3 Troubleshooting ABAP programs

Custom ABAP functions can cause a FUNCTION_CONFLICT runtime error executing generated ABAP if a parameter data type specified in the custom ABAP is not the same as expected by the SAP application.

For example, you write a custom ABAP program that writes a varchar(10) value to an SAP system variable, but SAP expects a char(30) value for the same variable. When you execute a job that contains a call to the custom ABAP program, the job fails at run time and adds a function conflict error to the error log for the execution.

4.4.2.5 Optimizing ABAP

Data Services generates ABAP code to extract data from SAP applications. Data Services optimizes the generated code to make the best use of SAP application resources. When optimized, the generated ABAP program executes as fast as possible with minimal impact on SAP applications. By producing the best possible ABAP program, Data Services minimizes execution time and maximizes total throughput.

Data Services makes use of the features offered by different version of the SAP platform. Therefore, the ABAP code Data Services generates depends on the version you are running.

4.4.2.5.1 Optimizations using SAP application open SQL features

Data Services optimizes ABAP using the open SQL features used by different versions of SAP applications. The open SQL features include:

- Inner joins
- Outer joins
- Aggregate functions
- Group by clause
- Order by clause
- Distinct statement

SQL features available in SAP applications

All supported versions of SAP applications support these open SQL features:

- Aggregate functions (AVG, MAX, MIN, SUM, COUNT*)
- Group by clause

- Order by clause
- Distinct statement
- Inner joins
- Outer joins

When possible, Data Services generates ABAP code that pushes these operations to the SAP database. To push the operation to the database, several conditions must be met:

- Query can be completed by a single ABAP SELECT statement
- Select list does not contain count() aggregate.
- No expression is inside an aggregate function.
- Parameter column of the aggregate function is a Data Services numeric data type.

  Note:
  Data Services numeric data types include integer, real, and double. The SAP NUMERIC data type is mapped to Data Services type VARCHAR, which is not a numeric data type.

When any of these conditions is not met, Data Services generates ABAP code that completes the operation within ABAP.

Optimizing inner joins

SAP applications can retrieve related data from multiple tables with a single SELECT statement using the JOIN command. You no longer need to use nested SELECT loops. Because a single SELECT statement requires fewer database accesses, performance improves.

When you create ABAP data flows that join transparent tables and that use at least one simple join predicate in the ON clause, Data Services creates a single SELECT statement using the JOIN command and pushes the operation to the SAP application database layer. For inner joins, simple join predicates are AND, =, <>, <, >, <=, and >=. These predicates must evaluate simple columns. Any complex join predicates are evaluated inside the select loop.

Data Services uses the table's join rank to determine which tables to join into a single SELECT statement and the order to join the tables. Specifically, tables that have the same join rank are joined together with a single SELECT statement that is pushed to SAP. Tables with higher join ranks are joined before tables with lower join ranks. Tables or groups of tables with different join ranks are joined with nested SELECT loops. These joins are not pushed to SAP.

When a table has the default join rank (zero), Data Services determines the best join rank for that table. Data Services decides based on which fields the table shares with other tables, using the fields that link the tables in the WHERE clause. Other tables in the join keep their assigned join rank.

When joining tables with a single SELECT statement, SAP determines which is the inner table and which is the outer table of the join. When joining tables with nested SELECT loops, Data Services uses the table or joined tables with the higher join rank as the outer table and the table or joined tables with the lower join rank as the inner table.

For example, suppose you want to join five transparent tables, A through E, with three different join ranks.

Table	Join Rank
A	10
B	10
C	5
D	3
E	3

Data Services creates a single SELECT statement joining A and B and pushes the operation to SAP (these tables have the same join rank and the highest value).

Next, Data Services creates a single select statement joining D and E and pushes the operation to SAP (these tables have the same join rank but a lower value). In both cases, SAP determines which table is the inner table and which table is the outer table.

Finally, Data Services creates a nested SELECT loop that joins the joined tables A and B with table C and joins the result with the joined tables D and E. The joined tables A and B are used as the outer table in the first join, and the joined tables A, B, and C are used as the outer table in the second join (the table with the higher rank is used as the outer table).

Now, suppose you specified four different ranks.

Table	Join Rank
A	10
B	4
C	5
D	3
E	3

In this case, Data Services creates a single SELECT statement joining D and E, and pushes the operation to SAP. Next, Data Services creates a nested SELECT loop that joins table A with table C, joins the result with table B, and joins that result with the joined tables D and E. Table A is the outer table in the first join, the joined tables A and C are the outer tables in the second join, and the joined tables A, C, and B are the outer tables in the third join.

Finally, suppose you specified five different ranks for the five tables.

Table	Join Rank
A	10
B	4
C	5
D	3
E	8

In this case, none of the joins are done with a single SELECT statement that is pushed to SAP. Instead, Data Services creates a nested SELECT loop. The loop joins A with E, the result with C, that result with B, and that result with D.

When Data Services determines which table is the outer table and which table is the inner table of a join, it uses information about the table size and estimates the effect any filters will have on the number of rows retrieved. Data Services uses the smaller data set as the outer table in a join.

Data Services determines the table size based on the **Estimated row count** attribute for the table. You can enter a value for any table.

For joins containing nested relations, Data Services uses the statistics of the embedding table when determining the best join order. For joins containing clustered fields, Data Services uses the statistics of the transparent table when determining the best join order.

Related Topics

- Specifying sources
- Optimizations using nested SELECT statements
- Determining the best join order
- Reference Guide: Table

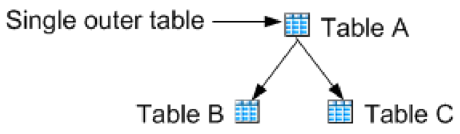
Optimizing outer joins

SAP applications can perform an outer join between two tables using the LEFT OUTER JOIN command. Using this ABAP command avoids awkward code that leads to unnecessary database table scans and poor performance.

SAP can only use the LEFT OUTER JOIN command when joining transparent tables with at least one simple expression in the ON clause. For left outer joins, simple expressions specify fields using AND commands with an equals condition. If the join also specifies a complex expression, Data Services will evaluate that expression inside the SELECT loop. Furthermore, SAP applications only support left outer joins when one table is used as the outer table in the join. In other words, within SAP you cannot have an outer join between two tables and use the result in another outer join.

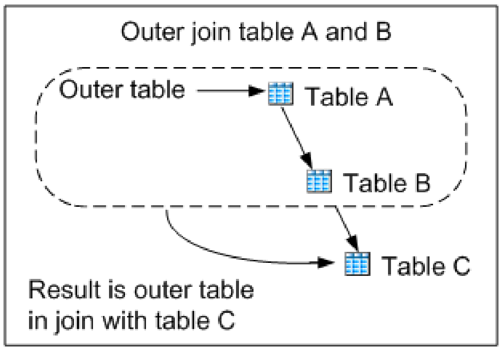
When you create ABAP data flows with an outer join that joins transparent tables and uses simple expressions in the ON clause, Data Services attempts to use the LEFT OUTER JOIN command and push the operation to the SAP application database layer.

When the outer join only uses one table as the outer table in the join, Data Services can create a single SELECT statement using the LEFT OUTER JOIN command and push the operation to the SAP application database layer.



You can use the join rank to control the order in which tables are joined.

When the outer join uses more than one table as the outer table in the join (such as the result of a previous outer join), then Data Services cannot create a single SELECT statement. Instead, Data Services chooses one outer join to complete using the LEFT OUTER JOIN command and then uses a nested SELECT loop to complete the join. Data Services randomly chooses which tables to join using the LEFT OUTER JOIN command.



4.4.2.5.2 Optimizations using nested SELECT statements

When working with SAP R/3 version 4.0 or later, you might need to use a nested SELECT statement because of restrictions on using the JOIN command and WHERE clause. When joining tables using a nested select statement, Data Services creates ABAP code that joins the tables in the order it anticipates is most efficient. You can override the decision with manual settings.

When joining data in a nested SELECT statement, Data Services extracts data in loops—the first table is read in an outer loop and subsequent tables are read in an inner loop. The number of loops depends on the number of tables joined. For best performance, you want the table that limits the most records read in the outer loop. By defining the smaller data set first, you will limit database operations when searching for matching records in the inner table. Data Services uses rules to ensure that this happens. You can override these rules and manually specify the join order.

During a join, only two tables are read at one time. However, outer joins that are split into multiple query transforms can reduce extraction performance. Therefore, join performance is better if outer joins that include more than two tables are specified in a single query transform.

Related Topics

• Designer Guide: Query transform overview

Automatically determining join order

Data Services automatically identifies the most appropriate table to read first in the join (that is, the table to use as the outer table). Data Services applies several rules:

1. Use the smaller source.

   Data Services estimates the table size using the Estimated_Row_Count attribute for the table. You can change this value to better reflect the actual size of the data set.

2. Use the table with fewer key fields specified in an ON clause.

   When joining tables, the ON clause specifies key fields that must match. Some tables, such as a detail table that is joined with a header table, have additional key fields that are not used in the match. Typically, the table with additional key fields is larger; the table with fewer key fields defines a smaller data set.

   Data Services detects the number of keys, and uses the table with fewer key fields in the outer SELECT loop and the table with the extra keys in the inner SELECT loop.

   Note:
   It is important to use all the SAP-specified keys when joining two related tables. If they are not included in the join, the underlying database used by the SAP application may scan the entire table rather than use the primary index.

3. Use the table with a constraint in the WHERE clause.

   When the WHERE clause in the joining Query transform includes a constraint, Data Services assumes that the selection based on a literal constraint will result in fewer selected records and that this table should be in the outer SELECT loop (which may not be the desired result).

For joins containing nested relations, Data Services uses the statistics of the embedding table when determining the best join order. For joins containing clustered fields, Data Services uses the statistics of the transparent table when determining the best join order.

Related Topics

• Reference Guide: Table

Manually setting join order

In some cases, you may not want Data Services to automatically determine the join order. For a particular join, the ON conditions or keys may lead to less-than-optimal performance when Data Services applies its rules to determine join order. In these cases, you can specify the join order manually.

To specify the join order manually, set the join rank to a positive non-zero integer for each table in the join. Set the join rank using the Query editor. The table with the largest join rank becomes the outer table in the join. The other tables are included based on the order of their join ranks.

If the join ranks of all source tables remain at the default value (0) and are also not specified in the Query editor, Data Services automatically assigns the join rank.

Even though the join ranks specify tables in the proper order, the underlying SAP application access methods might still prevent optimal performance. In this case, consider alternatives to the multiple table join, including using a two-way join with a table lookup using caching or small tables.

Related Topics

- Specifying sources
- Automatically determining join order

Determining the best join order

Use the information available for the data sets involved in a join to determine the best join order. This section shows two types of common joins and the techniques for determining the appropriate join orders.

Header and detail tables

For "header" and "detail" table joins, you usually want the header table to be the outer table and the detail table to be the inner table.

Because there are typically many line items for each header, the header table defines the smaller data set. Therefore, you want to use the header table as the outer table in the join and the line item table as the inner table in the join. The join operation reads all of the rows of the detail table into memory. Then for each row of the header table, the join matches multiple rows from the detail table. Producing the output requires reading through each table one time, with comparisons made for each row in the smaller table.

The alternative would be to read the header table first, then for each row of the detail table, read through the header table to find the matching details. Producing the output in this alternative requires reading through each table one time, with comparisons made for each row in the larger table. The cost of the join operation is higher in this case.

For example, when extracting rows from a purchase order line item table, you would join them with the corresponding row from the purchase order header table.

This join would produce a row for each controlling area and document number identified in the header table. You might limit the results by setting bounds on when the document was created, using document creation dates from the header table.

Identifier and description tables

When joining tables to extract text "descriptions" for "identifiers," you often find that the description table is much larger than the identifier table. In that case, you want to use the identifier table as the outer table and the description table as the inner table.

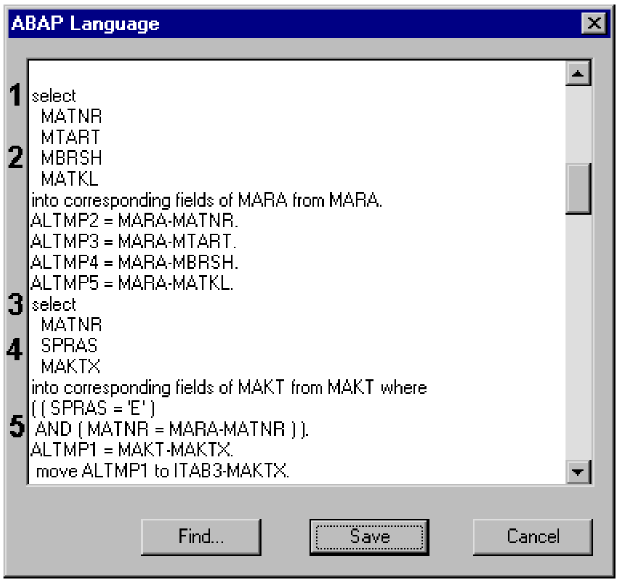
The join operation reads all of the rows of the description table into memory. Then for each row of the identifier table, the join matches a row from the description table. Producing the output requires reading through each table one time, with comparisons made for each row in the smaller table.

For example, when extracting business areas and descriptions, you might join areas from the business area table (the identifier) with text descriptions from the business areas description table (the description).

Because the description table is larger than the identifier table, you want the join to use the identifier table, TGSB, as the outer table and the description table, TGSBT, as the inner table. This join would produce a row for each business area identifier in the TGSB table. You might limit the results by language values from the TGSBT table.

Checking join order in ABAP

Before actually running a job, you can check the order in which tables will be joined by examining the ABAP that Data Services generates for the ABAP data flow. In the ABAP, look for the appropriate SELECT loop description. The outer table is described before the inner table. For example, for a join between two tables, MARA with join rank 2 and MAKT with join rank 1, the ABAP contains two SELECT statements indicating the outer table (MARA) and the inner table (MAKT).



```
1 select
   MATNR
   MTART
2  MBRSH
   MATKL
  into corresponding fields of MARA from MARA.
  ALTMP2 = MARA-MATNR.
  ALTMP3 = MARA-MTART.
  ALTMP4 = MARA-MBRSH.
  ALTMP5 = MARA-MATKL.
3 select
   MATNR
4  SPRAS
   MAKTX
  into corresponding fields of MAKT from MAKT where
  ( ( SPRAS = 'E' )
5  AND ( MATNR = MARA-MATNR ) ).
  ALTMP1 = MAKT-MAKTX.
   move ALTMP1 to ITAB3-MAKTX.
```

1. Outer SELECT statement
2. Column list from outer table
3. Inner SELECT statement
4. Column list from inner table
5. Join expression in the WHERE clause

To generate and view ABAP

1. Open the ABAP data flow in the workspace.
2. Choose **Debug** > **Generate ABAP** .

   The ABAP Language window opens in the workspace.

   If there are errors in the ABAP data flow, the Output window opens with a list of any errors or warnings. There cannot be any errors in the data flow for ABAP to be generated.

3. Locate the correct SELECT statements.

   Use the Find button or browse through the code by scrolling.

The first table in the set is the outermost table.

4.4.2.5.3 Optimizations for table caching

SAP applications can cache a table (that is, load it into memory). You can cache tables in two cases:

- Tables used in a join
- Tables used in a lookup function

Tables used in a join

Caching tables used as inner tables in joins improves performance as fewer database accesses are required. You can manually set or override table caching.

Note:
Best practice when joining multiple sources is to set join rank and cache for all sources in the Query editor.

When sources are joined using the Query transform, the cache setting in the Query transform takes precedence over the setting in the source. The default option in the Query editor is Automatic and in a table source is No. The automatic setting in the Query editor carries forward the setting from the source table. The following table shows the relationship between cache settings in the source, Query editor, and whether the software will load the data in the source table into cache.

Cache Setting in Source	Cache Setting in Query Editor	Effective Cache Setting in ABAP data flow
Yes	Automatic	Yes
No	Automatic	No
Automatic (Data Services determines whether or not to cache the table based on table size and number of filtered records)	Automatic	Automatic (based on table size and number of filtered records)
Yes	Yes	Yes
No	Yes	Yes
Automatic	Yes	Yes
Yes	No	No
No	No	No

Cache Setting in Source	Cache Setting in Query Editor	Effective Cache Setting in ABAP data flow
Automatic	No	No

Note:
If any one input schema has a cache specified in the Query editor, the Data Services Optimizer considers only Query editor cache settings and ignores all source editor cache settings.

The default option in the Query editor is Automatic and in a table source is No.

If you enable the cache on all tables in a join, inner and outer tables are read into internal tables prior to a join. This construct replaces SELECT LOOP processing. You should then use the Package Size option on the outer loop of the join to limit the size of the internal tables brought into memory.

The rows extracted from cached tables must fit into the production SAP system memory. When you select the **Automatic** option in the table source, Data Services estimates the table size and number of filtered records and projects the number of rows that must be extracted. When Data Services projects that 2,500 or fewer rows will be extracted and the table is used as the inner loop of a join, Data Services caches the table. To estimate the table size, Data Services uses the **Estimated row count** attribute for the table. You can enter a value for any table.

When cached tables do not fit into memory, Data Services throws an overflow or memory contention error when executing the ABAP program. Errors can occur when tables you select for caching (tables with the cache option set to **Yes**) do not fit into memory or when tables Data Services selects for caching (tables with the cache option set to **Automatic**) do not fit into memory.

If you experience either type of error, you have two options:

- Change the cache option to **No** for one or more tables, and re-run the program until you can execute it without an error.
- Use the Package size option to limit the number of rows read into memory.

Related Topics
- Specifying sources
- Reference Guide: Table

Tables used in a lookup function

Caching the values that a lookup function needs from a translation table reduces the number of times the system must be queried and improves system performance.

When creating a lookup function, you can specify whether or not to cache the translation table. When you specify that a translation table is cached, Data Services is able to improve performance. There are three options:

- **NO_CACHE**: Data Services caches no values.

- **PRE_LOAD_CACHE**: Data Services applies filters and caches all the matching lookup columns in the translation table.

  Select this option if the table can fit into memory.
- **DEMAND_LOAD_CACHE**: Data Services searches the ABAP data flow for other lookup functions that use the same translation table and have the same cache setting. When Data Services finds two or more lookups, it combines those lookups and caches the results into an internal table. The SAP application reads the internal table to find the results of individual lookups.

  Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

Because a lookup may retrieve many records, you must be careful that the caching requirement does not exceed available memory. If you get an error message indicating a lack of memory, you should remove caching from one or more tables.

Related Topics

- lookup

4.4.2.5.4 Making table size information available to jobs

When optimizing execution algorithms, Data Services makes decisions based on the size of tables being imported.

You can:

- Make table size information known to jobs for automatic tuning
- Use cost coefficients to further tune job performance

4.4.2.5.5 Optimizations for testing ABAP data flows

You can control the number of rows returned from an ABAP program when you run its job in test mode.

When you drop an ABAP data flow into the workspace, the Properties window for that data flow appears. On the **Options** tab, use **ABAP Row Limit** to limit the number of rows that the data flow reads. Then, select the **Test Mode** option from the "Execution Properties" window when you execute this job. This feature allows you to quickly run jobs in a development environment using SAP application data.

Related Topics

- Adding an ABAP data flow

4.4.3 Executing batch jobs that contain ABAP data flows

This section contains specific information about executing batch jobs that contain ABAP data flows.

Note:
This section assumes you are an experienced SAP user and have already followed similar procedures. Your system may be configured differently than the examples, so you should treat the steps in the procedures as high-level information—you may need to fill in additional details or you may need to do the steps in a somewhat different order.

Related Topics

- Designer Guide: Executing Jobs

4.4.3.1 Data Services functions supporting ABAP data flows

Using ABAP data flows, Data Services extracts data from SAP applications by directly accessing the SAP application server with Data Services-generated ABAP.

Data Services can execute the ABAP by the following methods:

- **Generate and Execute**

  This method calls the Data Services function /BODS/RFC_ABAP_INSTALL_AND_RUN. If this Data Services function was not uploaded to the SAP server, then the SAP function RFC_ABAP_INSTALL_AND_RUN is called.

  Note:
  This method requires that users who submit the job have S_DEVELOP authorization. Most SAP production systems do not provide this combination of authorizations to one user. Additionally, the RFC_ABAP_INSTALL_AND_RUN function is not well suited to execute in parallel. When multiple users access the system at the same time, some jobs could fail due to ABAP program name conflicts.

- **Execute Preloaded**

  This method uses Data Services-provided functions to overcome the security and stability issues inherent in the Generate and Execute method.

Related Topics

- ABAP execution options

4.4.3.2 Validating, generating, and installing ABAP code

Data Services enables you to check generated ABAP syntax errors before you run a job. The errors Data Services detects include mismatches between strings and numerals in parameter passing.

To request ABAP validation, you must have the Data Services function module /BODS/SYNTAX_CHECK installed on your SAP system. For information about the function modules and how to install them, see Installing Functions on the SAP Server.

4.4.3.2.1 To validate ABAP code created by Data Services

1. From the Designer, select an SAP data flow object (data flow, table, or transport) in the project area or open an object in the workspace.
2. Select **Debug** > **Validate** > **Validate ABAP**.

   Data Services checks for ABAP errors and opens a window to indicate what the errors are, or that there are no errors.

4.4.3.2.2 To generate and install ABAP independent of execution

To install generated ABAP code, you must first meet the following prerequisites:

- You must have necessary privileges in the SAP system to create a new program using transaction SE38.
- If the upload is performed while the SAP Applications datastore is in generate_and_execute mode, you must ensure that SAP client software (SAPGUI) is installed on the machine where the Data Services Designer is running.

Caution:
If a program already exists in the SAP system with the same name as an ABAP program you upload with this procedure, the uploaded program overwrites (without confirmation or warning) the previously existing program. You must ensure that name conflicts do not exist.

1. In the local object library, right-click an ABAP data flow object.

   Note:
   You can select more than one ABAP data flow object (even from different data stores) and proceed with these steps. However, you won't see the program of any of the data flows, as you do when you work with one data flow at a time.

2. From the context-sensitive menu, select **Generate ABAP Code**.

   The ABAP Programs Generation Dialog appears.
3. In the "Generate ABAP programs in directory" field, specify the path of the ABAP programs.
4. To enable automatic installation of the ABAP programs, select the **Upload generated programs** box.
5. Click **OK**.

   Data Services generates and installs the ABAP programs. Because the SAP system must process the ABAP programs, the installation time could be somewhat lengthy.

4.4.3.2.3 To change the Generated ABAP directory

1. In the object library, select an SAP Applications datastore.

2. Right-click and select **Edit**.
3. Click **Advanced**.
4. Enter the new directory path in the **Generated ABAP directory** box.
5. Click **OK**.

4.4.3.3 ABAP job execution scenarios

Much of the SAP Applications datastore definition determines how the ABAP data flow portion of a Data Services batch job executes and returns data.

4.4.3.3.1 ABAP execution options

For executing ABAP, Data Services provides the following options:

- **Generate and Execute**

  ABAP is sent to the SAP application server via the Data Services function /BODS/RFC_ABAP_INSTALL_AND_RUN. If this Data Services function does not exist, then the SAP-supplied RFC_ABAP_INSTALL_AND_RUN is used.

  This option is designed to be used during the development stage of a project. When a job is executed using an SAP Applications datastore set to the generate_and_execute mode, Data Services generates a random ABAP program name. Use the Designer's **Tools** > **Options** > **SAP** > **Environment** option called Prefix to ABAP program name to assign a prefix to the random name generated by Data Services. Generated files are stored in the directory you defined for the datastore.

- **Execute Preloaded**

  Communication between Data Services and SAP is handled by the Data Services RFC function modules, which must have been pre-installed on SAP.

  This option is generally used only in the test and production stages of an application project. When you select **Execute Preloaded**, Data Services doesn't generate ABAP to run the job. It simply calls an ABAP program name on the SAP application server. This program was created when you ran the job in Generate and Execute mode. Load the previously generated ABAP program into SAP, rename it as desired, and from the **Options** tab of the data flow Properties window, enter its name in the **ABAP program name** box.

4.4.3.3.2 Data transfer methods

ABAP data flows in Data Services allow you to automatically create ABAP programs. After an ABAP program runs, it creates a flat file (transport file) on the SAP application host. This file is later transferred from the SAP application host to a directory local to the job server executing the Data Services job.

To use a data transfer method for this outbound SAP file, select an option when configuring a datastore for your SAP application host. Data transfer options specified in an SAP Applications datastore editor determine the method by which the results (from batch jobs that contain ABAP data flows) are returned to Data Services. The options are:

- RFC: Use to stream data from the source SAP system directly to the Data Services data flow process using RFC.
- Direct download: SAP application server transfers the data directly to the **Client Download directory**.
- Shared directory: SAP application server loads the **Working Directory on SAP server** with the transport file. The file is read using the **Data Services Path to the shared directory**.
- FTP: SAP application server loads the **Working Directory on SAP server** with the transport file. The file is read using the **FTP relative path to the SAP working directory** and written to the **Client download directory**.
- Custom Transfer: SAP application server loads the **Working Directory on SAP server** with the transport file. The file is read by a third-party file transfer program and loaded to the **Local directory**.

Custom Transfer method

Using the custom transfer method, you specify a third-party file transfer program which Data Services uses to transport data between the remote SAP application server and a local directory on a Data Services Job Server machine. The transfer from SAP applications to Data Services is enabled using a data transport object (used in ABAP data flows).

When you select the Custom Transfer option, provide:

- A transfer program (to be invoked during job execution)
- (optional) Additional arguments used by the transfer program such as security, compression, or Data Services system variables.

Like other datastore settings, custom transfer program settings can be overridden if they are changed in a particular ABAP data flow. You can also edit the custom transfer option when exporting an SAP Applications datastore.

Datastore profiles support the custom transfer option.

Related Topics

- Designer Guide: File transfers
- Designer Guide: Creating and managing multiple datastore configurations

Data Services system variables to transfer a file from SAP applications

System variables can be used to provide the connection data you enter into Data Services to the custom transfer program. The following system variables can be used in the **Custom Transfer Arguments** box:

Data entered in this field:	Is substituted for this variable if it is defined in the Arguments box
SAP application server	$AW_SAP_HOST
Working directory on SAP server	$AW_SAP_WORK_DIR
Custom transfer user name	$AW_USER
Custom transfer password	$AW_PASSWORD
Local directory ("Local" denotes the job server computer)	$AW_LOCAL_DIR
File name This field is on the data transport object and is part of an ABAP data flow.	$AW_FILE_NAME

Direct download transport method

With **Direct download**, the data produced by the SAP application after running the ABAP program is returned by the Job Server computer via a ws_download function call. The data is stored in the **Client download directory**.

You should avoid using this method except when you need a quick way to return a small amount of data; it might not be reliable and does not scale well.

FTP transport method

The FTP method is most useful in heterogeneous environments (Windows and UNIX). It scales well and provides a good level of security, making it a recommended method of data transport.

Shared directory transport method

With shared directory transport method, both Data Services and SAP applications have direct access to the directory where data is stored. This method works best in a homogeneous environment (both systems are on Windows). The security risk inherent in a shared directory may pose a problem for some organizations. If this is the case, FTP is probably a better choice.

Shared directory requirements

To use the Shared Directory data transfer method, set up a shared directory between the SAP application server and your Data Services Job Server computer:

- If the SAP application server is on UNIX, use NFS or a similar disk sharing process. Use these steps to verify that the Job Server can access files generated by SAP:
  a. Log in to the SAP application server computer.
  b. Sign onto the NFS system.
  c. Navigate to the directory you specified in the datastore as the "Working directory on SAP server."
  d. Enter the command `touch bodi`.

     Note that you can use any term instead of bodi. The touch command creates a file with zero length in the current directory.
  e. Log into the Data Services Job Server computer.
  f. Navigate to the directory you specified in the datastore as the "Data Services path to the shared directory."
  g. Look for the file `bodi`.
  h. Use the command `filebodi` to verify that the Job Server can read the file.
- If the SAP application server is on Windows, use Windows directory sharing to provide mutual access to the staging area. Include paths to:
  - The shared device on the application server
  - The shared device on the Data Services Job Server

Note:
Because both the Data Services and the SAP application server can run on either Windows or UNIX platforms, mapping between computers can be interrupted if one of the computers is rebooted.

Observe the following recommendations:

- If you are running SAP and the Data Services on UNIX, make sure that your network will automatically mount the volumes you want to use when any computer is re-booted.
- If you are running the SAP application server on Windows and the Data Services Job Server on UNIX and you re-boot the UNIX computer, you may need to re-map Data Services to the SAP application server. If your jobs return error messages such as the following, disconnect then remap the NFS connection in SAP.

```
The SAP Job Canceled Host...
          File xxx cannot be opened...
```

Related Topics

• ABAP job execution scenarios

4.4.3.4 Sample execution sequence for an ABAP job

This section describes how a batch job that contains an ABAP data flow (also called an ABAP job) executes.

The following execution sequence is for a hypothetical job with the following characteristics:

- The batch job contains a data flow that, in turn, calls an ABAP data flow.
- The ABAP data flow extracts data from a subset of the columns in an SAP table.
- The calling data flow passes the output from the ABAP data flow into a query transform and loads a target table or file.
- The datastore options selected are **generate_and_execute** and **shared_directory**.

The scenario is valid for jobs initiated in Data Services as well as jobs initiated in SAP.

Oracle is assumed for this example, but the process is similar for other environments.

4.4.3.4.1 The execution steps

1. Data Services retrieves the job specification from the repository.
2. Data Services validates the job specification.
3. Data Services performs any required optimizations.
4. Data Services compiles the job specification.

   As part of this step, Data Services generates an ABAP program corresponding to the ABAP data flow specification and writes it to the path specified in the SAP Applications datastore options.

   All the parameters are replaced with data before ABAP is submitted to SAP to execute.
5. Data Services invokes `RFC_ABAP_INSTALL_AND_RUN` with the ABAP program as a parameter.
6. Depending on whether the ABAP job is running in the background or foreground, one of the following happens:
   - If the **Execute in background (batch)** option is set to **No**, then the ABAP program runs as a foreground job in dialog user mode in SAP. The `RFC_ABAP_INSTALL_AND_RUN` return code is checked for status. (`RFC_OK` return status implies successful completion of the ABAP program.)
   - If the **Execute in background (batch)** option is set to **Yes**, then the ABAP program runs in the background in SAP. A special Data Services-internal ABAP program runs in SAP and polls for status from the job status table TBTCO based on the job name that was assigned. The status column TBTCO-STATUS provides status information.
7. The ABAP portion of Data Services SAP-related jobs cannot run in parallel, so Data Services serializes the requested processes.

   Note:
   ABAP jobs can run in parallel when you specify execute_preloaded and the job executes with Data Services RFC.
8. Data Services schedules the job by calling SAP functions JOB_OPEN and JOB_CLOSE.
9. The job is scheduled as a class C job.
10. Data Services queries the job status every 15 seconds.

11. When the job finishes, the ABAP is deleted, but the job is still listed with a status of "finished."
12. When the ABAP program completes successfully, it writes output to the file name specified in the data transport in the path specified as **Working directory on SAP server** in the datastore definition.
13. The ABAP data flow is done and returns to the calling data flow.
14. Data Services reads the output file generated from ABAP data flow execution and processes it.

    Processing may include performing joins with other ABAP data flows or tables from other source databases or source files. It may also include any column transformations and/or transforms applied to the entire SAP output data set. Once all transformations are done, Data Services is ready to place the resulting data set in an output file or target table.
15. If the output is to a target table, these are the options available for loading the table:
    - If Oracle is the target database, then bulk loading is the typical choice.

      If bulk loading is turned on, Data Services generates the control file (.CTL) and the input (to the SQL*Loader) data file (generated_filename.dat) in the `C:/Data Services/Log/BulkLoader/` directory or the bulk loader directory specified in the Oracle datastore. It then invokes SQL*Loader on the client side (the system running Data Services) to load the data according to the control file specification.
    - If bulk loading is not used, Data Services builds and sends batch INSERT SQL statements to the Oracle server to execute and load the rows. A batch size option specifies the commit size.

Related Topics
- Execution modes of ABAP data flows
- Debugging and testing ABAP jobs
- Performance Optimization Guide: Using Bulk Loading

4.4.3.5 Scheduling jobs from SAP

The following is an overview of the execution process for jobs initiated in SAP.
1. When it is time to run a job, SAP sends an extract request to Data Services through the RFC server.
2. Data Services generates ABAP and submits it to SAP.
3. The Data Services batch file initiates execution. The job runs on the job server specified when the job was exported.
4. The ABAP is executed on the SAP application server.
5. The SAP application server generates the specified results.
6. Data Services is notified of the results.

4.4.3.5.1 Overview of preparation steps

For jobs where the schedule is set and the job is initiated by SAP, do the following.

1. In Data Services, export the job to a batch file (*.bat file for Windows environments, and *.sh file for UNIX environments).
2. In SAP from transaction SE38, create the ABAP program that will execute the job schedule.
3. From transaction SM36, create the job schedule.
4. From transaction SM59, create the RFC destination by registering the Data Services RFC server with SAP.
5. In Data Services, ensure that the Job Server is running (or will be running when the job begins to execute).
6. Add an RFC Server Interface.
7. Start the RFC server or ensure that it is running (or will be running when the job begins to execute).

In SAP from transaction SM37, you can monitor the job as it runs. You can also monitor the job in Data Services. The Designer does not need to be running. If it is running, the execution window opens when the job begins to run.

Related Topics

• Management Console Guide: Administrator, Batch Jobs, Using a third-party scheduler (To export a job for scheduling)
• Designer Guide: Executing Jobs, Ensuring that the Job Server is running
• Monitoring ABAP jobs in SAP applications

4.4.3.5.2 Creating the program that will execute the schedule

You need to create the ABAP program that will initiate communications with the Data Services RFC server to execute the job schedule.

We recommend that you use the model ABAP code provided for you at `Data Services\Admin\R3_Functions\schedule\yschedul.abap` (the procedures in this section assume you do so). You can also create your own program.

4.4.3.5.3 To create an ABAP program in SAP applications

1. Run transaction **SE38**.
2. Select **yschedul** in the **Program** drop-down box.
3. Click the **Create** button.

   The Program Attributes screen appears.
4. From the Program Attributes screen:
   - In the **Title** box, add a title.
   - In the **Type** box, enter **1** (online).
5. Click **Save**.
6. Click the **Source code** button.

7. In the **Create object catalog entry** screen, enter the development class for this job.

   This can be any development class you have created for this purpose. We recommend using zact.
8. Click **Save**.

   An ID number has been assigned to this job (the correction and transport number).
9. Click **Save**.

   The ABAP/4 Editor: Edit Program screen appears.
10. Choose **Upload** from the **Utilities** > **Upload/Download** menu.
11. From the Upload dialog, navigate to yschedul.abap.

    The default location is `C:\Data Services\Admin\R3_Functions\Schedule`.
12. Click **OK**.

    This returns you to the ABAP/4 Editor: Edit Program screen.
13. Click **Save**.
14. Click the back arrow to get back to the ABAP/4 Editor: Initial Screen.
15. From the ABAP/4 Editor: Initial Screen, click **Execute**.
16. Add the fully qualified path to the batch file containing the job you want to run and the RFC destination name.
17. Click **Save**.
18. Enter a variant name and (optionally) a description.

    A variant name is a collection of parameters that determine how a job is run. Use any name that is meaningful to you.
19. Click or press **Enter**.

    The SE38 procedure is complete.

4.4.3.5.4 To create a job schedule in SAP applications

1. Run transaction **SM36**.

   View the Define Background Job screen.
2. From the Define Background Job screen:
   a. Add a job name.
   b. Add the target host (the SAP application server).
3. Click the **Steps** button.

   The Create Step 1 screen appears.
4. In the Create Step 1 screen, click the **ABAP/4** button.

   Items in the ABAP program section are now available.
5. Add the program **Name** (yschedul).
6. Add the appropriate variant information.
7. Click **Save**.
8. Return to the **Define Background Job** screen.

9. Click the **Start date** button.
10. Add the date and time for the job to run.
11. Complete additional information as needed and click **Save** to complete the job schedule.

The job runs at the time you scheduled it.

If Data Services is active when the job begins to run, the execution window (with the trace button active) appears.

4.4.3.5.5 To register the Data Services RFC server in SAP applications

1. Run transaction **SM59**.
2. If you have not yet created the TCP/IP connection between Data Services and SAP applications, do so from the TCP/IP connection choice on this screen.
3. When the TCP/IP connection has been created and is active, click the **Create** button on the Display and Maintain RFC Destinations screen.
4. In the RFC destination box, add an RFC destination name.

   This can be any name you choose.
5. Under **Technical Settings** > **Connection type**, choose **T** (for TCP/IP).
6. (Optional) Under **Description**, add descriptive text.
7. Click or press **Enter**.

   The RFC Destination screen appears.
8. In the **Registration — Program ID** text box, fill in the name of the Program ID.

   This is the job name you specified for RFC Program name in the RFC Server Interface connection (see Starting the RFC server).
9. Click the **Test connection** button to check if the connection is successful.

   If the connection is not successful, you receive an error message. You need to debug the problem before proceeding.

4.4.3.5.6 Starting the RFC server

The Data Services RFC (remote function call) server is the execution link for Data Services jobs initiated by SAP applications or SAP NetWeaver BW.

To start the RFC server, in the Administrator from the **RFC Server Interface Status** page, select the desired RFC server and click **Start**. For details, see the *Data Services Administrators Guide*.

You can verify the RFC server is running in several ways:

- In the Administrator, select **SAP Connections** > **RFC Server Interface** > **RFC Server Interface Status** tab.
- View the SAP application processes.
- View the Windows or UNIX processes (depending on where the RFC server has been installed in your environment)

  If the Data Services RFC server is running, it appears among the running processes listed, for example RfcSvr.

Related Topics
- Management Console Guide: Starting or stopping an RFC server interface connection

4.4.3.6 Monitoring ABAP jobs in SAP applications

You can monitor an ABAP job generated by Data Services just as you would any ABAP job using the SM37 batch monitoring window.

The Data Services job name appears in the list of jobs in the SM37 window.

4.4.3.7 Execution modes of ABAP data flows

There are two execution modes for ABAP data flows:
- Dialog execution mode
- Batch execution mode

Note:
You cannot use the dialog and batch capabilities at the same time.

Related Topics
- Dialog execution mode
- Batch execution mode

4.4.3.7.1 Dialog execution mode

In dialog execution mode, Data Services connects to SAP, submits and executes an ABAP program as a foreground job in SAP, then disconnects from SAP after the program execution finishes.

Dialog is the default execution mode.

An ABAP program executed in dialog mode is subject to the dialog execution time constraint (typically set to be 5 minutes) defined in the SAP system profile. If the execution time of a program executing in dialog mode exceeds the specified time-out period, SAP aborts execution of the program, producing an ABAP data flow runtime error and an exception error in Data Services.

The execution time constraint is set using the vdisp/max_wprun_time parameter in the profile corresponding to the system executing the job.

4.4.3.7.2 Batch execution mode

In batch execution mode, Data Services connects to SAP, submits the ABAP program as a background job in SAP, then disconnects from SAP after the program is submitted. Data Services schedules and executes the submitted program as a background job.

An ABAP program executed in batch mode is not subject to the dialog execution time constraint mentioned in that mode. However, there is a slight performance overhead due to background scheduling in SAP and execution status checking by Data Services.

Note:
If you kill a Data Services job containing an ABAP data flow defined to run in batch mode, the batch program still executes. You must use SAP facilities to terminate the batch program.

When more than one ABAP data flow is running concurrently in batch mode, the batch job submissions are serialized. However, after the batch jobs are submitted, they run in parallel in the SAP application server if the server has more than one batch processor available.

To run the ABAP generated by the ABAP data flow in batch execution mode, click **Advanced** on the Edit Datastore window and set the **Execute in Background (batch)** option to **Yes**. Make sure that the Data Services SAP functions are uploaded to the SAP system running the job.

Note:
The SAP application server name is not case sensitive, but the **Target host** name must be entered as it is actually registered in the SAP system.

Related Topics

• Installing Functions on the SAP Server

4.4.3.8 Debugging and testing ABAP jobs

Generating and executing ABAP programs from Data Services can result in the following kinds of errors:

Task	Type of error	Result
Design	Validation errors	Execution stops. Information is available interactively (in a pop-up error window).
ABAP generation	Generation errors	Execution stops. Information is available interactively (in a pop-up error window).

Task	Type of error	Result
ABAP validation	ABAP syntax errors	Execution stops. For errors detected by Data Services, detailed information is provided in the error log. For errors detected by SAP, the message in the error log is RFC_CLOSED. In this case, you need to run the ABAP program manually in SAP to identify the problem more precisely.
ABAP execution (dialog mode)	ABAP execution errors	Execution stops. Information is available interactively (in a pop-up error window).
ABAP execution (batch mode)	Job execution errors (job was cancelled)	Execution stops. The job log from SAP is available in the Data Services error log.
ABAP execution	Function exceptions	Causes an exception. Information is available in the Data Services error log if the Data Services function modules have been installed on SAP.

Related Topics

• Installing Functions on the SAP Server

4.4.3.9 Generating ABAP code

You can generate ABAP code to be executed manually on SAP.

4.4.3.9.1 To generate ABAP code

1. Open the ABAP data flow.
2. Choose **Validation** > **Generate ABAP code**.

   When the generation is complete, Data Services opens the text file containing the ABAP program.

   This file name and location were specified when you defined the ABAP data flow.

4.5 Working with functions

You can use SAP remote function calls (RFCs) in queries created in Data Services data flows. In addition, Data Services provides the SAP application BAPI interface to support the use of remote function calls designed for business transactions (BAPIs). Use the SAP application datastore to import BAPI function metadata.

SAP functions that are not RFC-enabled can be used in ABAP data flows with the following restrictions:

- The function can have only scalar, multiple input parameters. The function cannot use table parameters.
- For the output you can select only one scalar parameter.

Data Services cannot use normal functions in data flows because an SAP application normal function is not an RFC function.

However, you can write a wrapper for a normal function, which can change the function to be RFC enabled. In this case, a normal function would be supported in data flows including table parameters.

Related Topics

- SAP application RFC and BAPI function calls

4.5.1 Calling BAPI functions

A real-time job can apply data to SAP applications using a real-time job remote function call (RFC). The RFCs that are included in SAP application releases, called Business Application Programming Interface (BAPI) function calls (such as BAPI_SALESORDER_CREATEFROMDAT1) can be particularly useful.

The metadata for the function—displayed through the function wizard in a query—describes the input and output parameters of the function, including the columns and tables included in the transaction applied to SAP.

Related Topics

- Browsing, searching, and importing metadata from SAP applications
- Calling RFC functions

4.5.2 Calling RFC functions

A real-time job can use RFC-enabled functions to retrieve information from and apply information to SAP applications. With nested tables, you can include tables as input or output parameters to the function. Functions allow you to:

- Return a specific response based on specific input you provide to the function
- Apply data to or retrieve data from more than one SAP table at a time

RFC functions can require input values for some parameters; SAP supplies default values for other inputs, and some can be left unspecified. You must determine the requirements of the function to prepare the appropriate inputs.

Note:
To avoid returning errors from RFC calls, make sure input is formatted as required by SAP. For example, all character data must be in uppercase; some values require padding to fill out the length of the data type.

To make up the input, you can specify the top-level table, top-level columns, and any tables nested one-level down relative to the tables listed in the FROM clause of the context calling the function. If the RFC includes a structure as an input parameter, you must specify the individual columns that make up the structure.

A data flow may contain several steps that call a function, retrieve results, then shape the results into the columns and tables required for a response.

4.5.2.1 To call a function that includes tables as input or output parameters

1. Create a table that includes the columns required as input to the function as the input data set to a query.
2. In the output schema of the query, right-click and choose **New Function Call**.
3. In the "Select function" wizard, under "Function categories" select the datastore corresponding to the SAP application server where you want to apply the function.
4. Select the RFC function to call and click **Next**.
5. Enter the input parameters for the function and click **Next**.

   If there are more parameters than fit on a page in the function wizard, click the down arrow to display the next page of input parameters.
6. Select which parameters to return by selecting a column or table to include on the left and moving it to the list on the right. Click **Finish**.

   In the Query, the function appears as a nested table in the output. Your output parameters are listed as columns or tables under the function.

7. The function is automatically marked to unnest in the output of the query, so the nesting level indicated by the function name is not included in the input to the next step in the data flow.

4.6 IDoc interface connectivity

Your SAP system should be configured to send and receive IDocs from Data Services. Refer to the SAP application documentation for more information. Set the communication type to Unicode if the SAP system supports it.

IDoc message sources (SAP outbound IDocs) must be configured in the Data Services Designer and in the RFC client branch of the Data Services Administrator. However, you do not need to configure RFC clients for IDoc message targets (SAP inbound IDocs). The partner number, partner type, message type, and IDoc type settings you save in the Designer's IDoc target editors are the ones needed to send messages to SAP application via transactional remote function calls (tRFCs).

The remainder of this section describes Data Services functions and SAP security levels and user authorizations. These sections all apply to the ABAP, RFC, and IDoc interfaces.

Related Topics

- Source
- Importing IDoc metadata using the Search option

4.6.1 Batch and real-time processing overview

An SAP IDoc is an intermediate document that can be used to send and receive business documents between SAP application servers and other servers.

SAP applications offer numerous IDocs that describe different business documents. The data in an IDoc supports a particular transaction. An IDoc can contain data from multiple tables and can contain hierarchical data.

Data Services processes outbound and inbound IDocs from and to the SAP application server. SAP outbound IDocs are called IDoc sources in Data Services. SAP inbound IDocs are called IDoc targets.

The method of processing IDocs depends on the type of job in which you define the IDoc and whether the IDoc is handled as a message or a file.

In Data Services, an IDoc can be used as a source or target object in a batch or real-time job. An Idoc can be processed as a message or as a file, however the only limitation is that batch jobs cannot process IDoc messages as sources.

For example, Data Services real-time processes IDoc source messages (outbound IDocs from the SAP application server) as follows:

- The SAP system administrator includes the Access Server as a TCP, remote function call (RFC) destination.
- The SAP application server (client) publishes an IDoc after processing a transaction. For example, when an operator enters a sales order.
- The Access Server receives the published IDoc and dispatches it to an available service provider for processing.
- If no service providers are available for this IDoc type, the Access Server queues the request and dispatches it when a service provider becomes available.
- When available, a service provider processes the IDoc. The IDoc data can trigger queries to the SAP application server or cached data as required to enrich the IDoc data. The real-time job flattens and loads the required information into a data cache and/or an IDoc message target as specified in its data flow design.
- The service provider returns a response to the Access Server indicating the successful IDoc processing.
- The Access Server receives the response and sends an acknowledgment to the SAP application server so the IDoc can be marked as processed.

Related Topics

- IDoc interface connectivity
- Browsing, searching, and importing metadata from SAP applications
- IDoc file
- IDoc message
- Source
- Target

4.6.2 IDocs and batch jobs

4.6.2.1 IDoc sources in batch jobs

In batch job data flows, IDoc file sources can hold one file or many individual files (to be processed at a later time). See Source for the list of IDoc source editor options.

IDoc file sources are processed as follows:

- The IDoc file is opened and the IDoc record type is examined to see which type it is.

Note:
IDoc file sources are validated against the IDoc Type, the IDoc Type related hierarchy, and the SAP Release.

- Related IDoc metadata is read.
- The IDoc list (one or many IDocs) is built based on the information in the **IDoc File** box.

  If an error occurs, a message like the following one displays:

  ```
  Cannot build IDoc from IDoc file: <%s>. Please see the error
      log.
  ```

  The process terminates, and the next file is processed (if multiple IDoc files are defined).
- Each IDoc is read one by one from the IDoc list and passed to the query. A trace message is recorded in the job's trace log such as:

  ```
  Process IDoc number: <no. in the file: filename>.
  ```

4.6.2.1.1 Multiple file read

You can configure Data Services to read multiple files by listing the names of these IDoc files in the **IDoc file** box of the source editor in the Designer.

Examples:

- If you specify the IDoc file as D:\temp\IDOC_R3*.txt, Data Services processes data from all files in the D:\temp directory with names beginning with IDOC_R3 and the extension .txt.
- If you specify the IDoc file D:\temp\IDOC_R3?.txt, Data Services processes data from all files in the D:\temp directory with names beginning with IDOC_R3, with any character before the extension, and the extension .txt.
- If you specify the IDoc file D:\temp\IDOC_01.txt, D:\temp\DOC_02.txt, D:\temp\IDOC_03.txt, Data Services processes data from these three IDoc files.

4.6.2.1.2 Variable file names

You can use variable file names in the **IDoc File/s** box. Define a global or local variable for an initialization script of a job. For example:

```
$filepath = 'D:\temp\IDOC_R3*.txt'
```

Then enter $filepath in the **IDoc File/s** box. In this way you can avoid opening each IDoc object to configure the location of source files.

4.6.2.2 IDoc targets in batch jobs

In batch job data flows, one or more sources can be transformed into an IDoc message target.

The sources may become active on a scheduled basis. As soon as all sources are present, the transform (a query, for example) processes the data into the IDoc format configured in the query's output schema and the IDoc message(s) are sent to SAP via a tRFC connection.

A data flow that contains an IDoc file target can be used to store IDocs. When you configure an IDoc file target, Data Services allows you to capture data for more than one IDoc. Each time a root segment passes into a file target (for example, **IDoc File** C:\temp\idoc), a new file is created and named using an eight-character suffix starting with 00000000.txt. For example, an **IDoc File** set to C:\temp\idoc can store idoc00000000.txt to idoc99999999.txt.

See Target for a list of IDoc target attributes.

4.6.2.3 Adding an IDoc to a batch job

When you drag an IDoc into a data flow, Data Services displays a popup menu, with valid options: **Make IDoc file source**, **Make IDoc file target**, **Make IDoc message target**.

To add an IDoc to a batch job:

1. Select an IDoc from the object library.
2. Drag it into an open data flow.
3. Select the appropriate option.

4.6.3 IDocs and real-time jobs

4.6.3.1 IDoc sources in real-time jobs

For real-time data extraction from and loading to SAP applications, Data Services uses SAP Application Link Enabling (ALE) technology and Intermediate Documents (IDocs) to capture and process transactions

Each real-time job can contain only one message source. For extracting data from SAP applications, using Data Services, that message source must be an IDoc.

A real-time job can contain other sources. Data from these supplementary sources is extracted on-demand when the real-time job is processing data from the message source.

The message source object represents the schema of the expected messages. Messages received are compared to the schema. If a data flow receives a message that does not match the schema, the data flow throws an error at run time.

You must import the schema for each SAP IDoc message type your real-time system will process. See Defining SAP Applications datastores for information about creating a datastore and importing metadata for IDocs.

4.6.3.1.1 Creating real-time jobs using IDoc sources

To create and run a real-time job that uses an IDoc message source, perform the following steps.

First see IDoc interface connectivity for information about configuring SAP to send IDocs to Data Services.

In the Designer:

1. Create an SAP Applications datastore.
2. Import IDoc metadata into the Data Services repository.
3. Create a real-time job using one or more data flows.
4. Drag an IDoc Type into the first data flow and select **Make IDoc message source**.

   Note:
   You cannot configure XML message sources or targets in a real-time job that has an IDoc message source.

5. Add a query and a target to the data flow.
6. Connect the objects.
7. Map the schema using the Query editor.
8. Validate and execute the real-time job by running it in test mode. Results are saved to a file you specify.
9. When the job runs as desired, open the Data Services Administrator to set up the job in a test/production environment.

In the Administrator:

1. If you have not already done so, add a connection to the repository that contains the real-time job and add a connection to the Access Server that will manage the real-time service.
2. Add a service and service providers for the real-time job.
3. Configure an Access Server to communicate with SAP to process IDocs in real-time jobs as follows. Under the **Real-Time** node for the Access Server to configure, click **Client Interfaces**. From the **Client Interface Configuration** tab, add an RFC connection, then add supported IDocs from the **List of Supported IDocs** tab. On the **Add IDoc** page, specify the **IDoc type** (used in the job) and the **Service name** (that you just created for the real-time job).
4. From the **Real-Time Services Status** page, start the service for the real-time job.
5. From the **Client Interface Status** page, start the client.
6. From the **Real-Time Services Status** page, monitor the service and view Job Server error and trace logs to verify the service is successfully processing IDocs from SAP.

4.6.3.1.2 Request/acknowledge data flow model

The request/acknowledge data flow model supports the use of an IDoc as a message source in the first data flow inside a real-time processing loop. Data Services automatically generates a final data flow in any real-time processing loop that contains an IDoc message source. This final data flow contains an XML message target.

The XML message target sends an acknowledgement back to the Access Server indicating that the IDoc message source was received from the SAP application. The Access Server returns the acknowledgement to the SAP application.

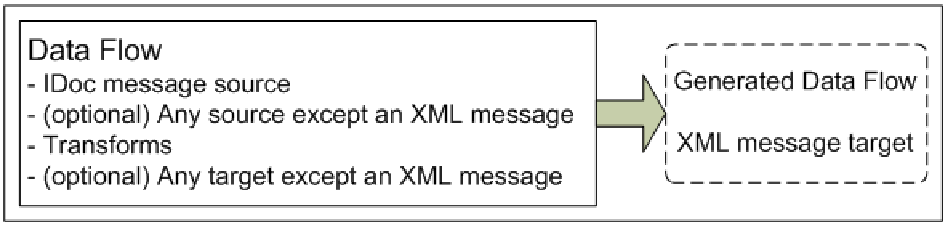

Data Flow
- IDoc message source
- (optional) Any source except an XML message
- Transforms
- (optional) Any target except an XML message
Generated Data Flow
XML message target

Figure 4-1: Request/Acknowledge model with a single data flow

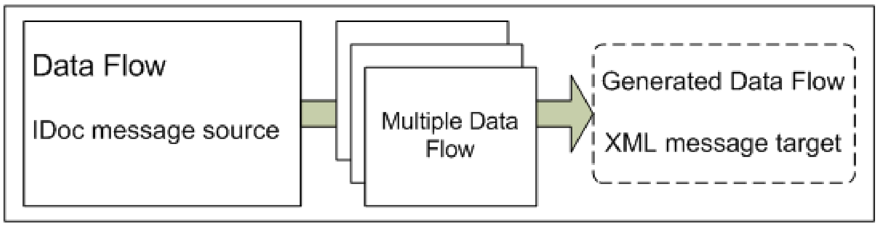
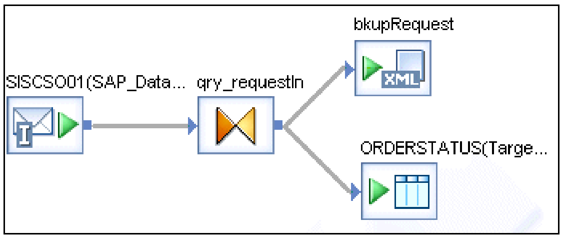


Figure 4-2: Request/Acknowledge model with multiple data flows

When using the request/acknowledge model, after the first data flow you can configure additional data flows, work flows, scripts, conditionals, and so on. The only limitation is that because an IDoc message source is used, XML message sources and targets cannot be configured in the job. Outside the real-time processing loop you can add initialization and/or clean-up objects as desired.

First data flow in a request/acknowledge model

In a request/acknowledge data flow model, the first data flow in a real-time processing loop contains an IDoc message source. To move a data set to succeeding data flows (optional), you can use a memory table as shown in the example below.



4.6.3.1.3 Adding an IDoc source to a data flow

Use the following procedure to add an IDoc source to a data flow.

1. Open the data flow in the workspace.
2. In the object library, select the **Datastores** tab.
3. Expand the datastore that defines the connection to the SAP application server.
4. Inside the datastore, expand the IDocs node.

5. Drag the IDoc that the data flow will process into the workspace and choose to use it as a file or message source by selecting **Make IDoc file source** or **Make IDoc message source** from the pop-up menu.

   Note:
   If your data flow is the first one inside a real-time processing loop, it can have one and only one message source. If another message source already exists in the data flow definition, Data Services will not allow you to place another one in the workspace.

6. Drop a Query and a target (optional if your data flow is part of a real-time job) and connect the objects.
7. Define source editor options by clicking the name of the IDoc to open the source editor.

   If the IDoc source contains nested data, the schema displays nested tables to represent the relationships among the data.

   IDocs contain many nested tables and potentially hundreds of columns across the levels. It is possible that you only need to load data from a fraction of the columns from the IDoc. To make working with the IDoc contents easier, consider deleting columns and nested tables that you do not need.

8. Define the output schema.

   First double-click the Query object to open it.

   You can drag and drop or copy and paste columns from the input schema to the output schema. The icons indicate when a column is mapped. The following table shows sample input and output schemas:

Input schema	Description	Output schema	Description
E1CVBAK-VBELN	Sales document number	E1CVBAK-VBELN	Sales document number
E1CVBAK-AUART	Sales document type	E1CVBAK-E1CVBP0-PARVW	Partner function
E1CVBAK-E1CVBAP-POSNR	Sales document item number		
E1CVBAK-E1CVBP0-PARVW	Partner function		

   Considerations:

   - All IDoc fields have type char.
   - To use two columns with the same name, you must change the name of the first column before adding the second column.

9. (Optional) Join IDoc data with an SAP table using a Query transform.

   Extract data from the SAP application table using an RFC function call. This is efficient for a small amount of data.

Related Topics
- Source
- IDoc file
- IDoc message

4.6.3.2 IDoc targets in real-time jobs

Most actions that users perform through a Web-enabled application require some response from the system in the form of data. The response can be as simple as an acknowledgment that a message was received. It can also be a specific set of data extracted from an SAP system or data cache. If you have configured a real-time job using an IDoc message source, you do not need to configure a target to send an acknowledgement because Data Services generates an acknowledgement (an XML message) automatically.

IDocs can also be configured as message or file targets in real-time jobs. IDoc message targets move data to SAP applications. IDoc file targets can be used to collect data for testing, data flow validation, and backup.

For example, you might want to create two identical data flows. The first has an IDoc message target and the second has an IDoc file target. The second data flow could be used to store IDocs in a file temporarily. It could be used in a job that your system administrator starts only when it is known that the target server will be down. Later, you can use a job to update the database with the data captured in the IDoc file target.

IDoc target messages can be sent from Data Services on a regular schedule or in real-time because you can choose to use IDoc message targets in either batch or a real-time jobs.

While there can only be one IDoc message source in a real-time job, any number of IDoc targets can be included in a data flow. In addition, if your source is not an SAP application server, a data flow can have an XML message as a source and an IDoc message or file as a target.

Any number of IDoc message targets can be added to a data flow. A data flow can also contain other targets such as a table or file.

Data flows in real-time and batch jobs that contain an IDoc message target send IDocs to SAP applications in the same way. The only difference is when the data is sent. Data can be sent to SAP applications on demand, or if you use an IDoc message target in a batch job, data is sent when the job is scheduled to run.

When a job is executed, output statistics from the IDoc target object are written to the real-time job's monitor log. The number of IDocs written is recorded. The number of IDocs increments for every root-level row of data passed to SAP because IDocs contain only a single root segment.

4.6.3.2.1 Data and format considerations

IDoc targets can insert and update data in SAP applications. For example, to create an IDoc target in Data Services, you import metadata from an existing IDoc, then reuse this metadata to create an IDoc target object in Data Services.

Data

The following examples list data consistency issues you should consider when planning to use IDoc targets.

Example 1

If your IDoc contains information to update a sales order, then the target IDoc (sales order) must:

- Be defined with a valid IDoc Type (name).

  In SAP, a name for an IDoc is called an IDoc type. IDoc types are subsets of message types. An example of a message type is ORDERSP, while an example of an IDoc type is ORDERS04.
- Match the SAP order number.
- Match SAP segment field formatting.
- Contain all mandatory IDoc segments for its message type.

  SAP segments are sections of IDocs. They are equivalent to tables and nested tables in Data Services. Segment fields are equivalent to columns. Each SAP message type has required segments and segment fields without which the IDoc will not be posted to SAP.

  The Query Transform Definition window in Data Services displays SAP segments (as tables) and SAP segment fields (as columns with data types). However, the Query Transform Definition window does not indicate which segments are required, nor does Data Services enforce SAP's segment repetition rules. An attribute of a segment is that it has a minimum and maximum number of times that it can occur in a single IDoc.

View SAP front-end applications to ensure you have the required order numbers and segments and your IDoc will not violate segment repetition rules. This information is also available in Data Services in the object's class attributes metadata.

Example 2

If your IDoc is a new sales order, then the sales order must:

- Be defined with valid a IDoc type, partner number, partner type, and message type names.

  Partner numbers and types are used to distribute IDoc data in SAP applications. If they do not exist or you do not specify the correct values, your IDoc will not post to SAP successfully. Use SAP front-end applications to configure SAP to receive an inbound IDoc (IDoc target) from Data Services.

  SAP applications define message types for different IDoc tasks. The message type you choose must be appropriate for the processing goal of the IDoc. Message types provide part of the required description of what is being passed to SAP. Again, reference the list of message types using SAP front-end applications.
- Match SAP segment field formatting.

- Contain all mandatory IDoc segments for its message type.

Data types and formats

Data Services supports several data types. IDoc Targets support only varchar.

- If an IDoc Target has columns configured in the varchar data type, no conversion occurs when it is sent to an SAP application.
- If an IDoc Target column uses another data type, the data type is converted to varchar and a default ABAP format.

The following table lists Data Services data types and their default formats:

Data Services data type	Is converted to ABAP...
date	date (YYYYMMDD)
datetime	date (YYYYMMDD)
decimal	float
double	float
int	int
interval	float
long	int
real	float
time	time (HHMISS)
varchar	no conversion

These default formats are defined by Data Services to help with the data movement to SAP applications. However, if the IDocs or the ABAP modules for IDoc processing in SAP are customized, you are responsible for modifying the default formats as needed to match the requirements.

- An IDoc can be customized in SAP

  Importing the metadata for your IDoc target can limit the risk of having a failed posting to SAP due to inconsistent formats.

- Input and posting modules can be customized

  IDocs are generated and read by ABAP modules. IDocs are processed in SAP applications first by an input module and then by a posting module. ABAP program modules control how data is interpreted. The configuration of these modules can be customized.

  For example, a posting module might be coded to receive a date formatted such as MMDDYYYY or DDMMYYYY instead of the standard ABAP date type format YYYYMMDD.

You must modify the metadata in Data Services or the format of the IDoc data generated in SAP to account for customized formats.

4.6.3.2.2 Adding an IDoc target to a data flow

When you drag an IDoc into a data flow, Data Services presents a pop-up menu of options for the type of data flow you are using.

To add an IDoc target to a data flow:

1. Select an IDoc from the object library.
2. Drag it into an open real-time or batch data flow.
3. Select **Make IDoc file target** or **Make IDoc message target**.
4. Add one or more source objects and a Query.

   You must copy imported IDoc file and message targets from the IDoc message target definition window into the target schema of a Query before using them in Data Services data flows.
5. Connect the source object(s) to the Query and the Query to the IDoc target object.
6. Click the name of the IDoc target.
7. Complete the IDoc target editor boxes. See Target for more information.
8. Click **OK**.

4.7 File formats

This section discusses SAP application file formats and how they relate to Data Services.

Related Topics

• Designer Guide: File Formats

4.7.1 The Transport_Format

The Transport_Format is a Data Services predefined file format object for reading flat files from SAP applications. It is automatically used to define data transport objects added to SAP application data flows.

The Transport_Format appears in the **Format** tab of the object library under **Flat Files**.

You can modify it for your own use. You cannot delete it.

4.7.1.1 To edit the Transport_Format

1. From the object library, right-click Transport_Format and select **Edit**.

   The file format editor opens.

2. Notice that the "Type" is set to **SAP transport**, the "Name" is set to **Transport_Format**, and the "Delimiters", which indicates where one column ends and another begins, uses the ASCII characters /127 as the column marker.

   The default delimiter for the Data Transport object is /127.

4.7.2 Defining an SAP applications file format

When you want to read from or write to a file in an SAP data flow and do not want to use the predefined Transport_Format, you can create your own SAP application file format.

1. Open the object library.
2. Click the **Formats** tab.
3. Right-click **Flat Files** and choose **New**.

   The file format editor opens.

4. Set the **Type** to **SAP transport**.
5. For the **Name**, use the automatically generated name, or enter a name to identify the format inside Data Services.

   The name can include alphanumeric characters and underscores (_). It cannot contain blank spaces.

6. (Optional) To use an existing file as a template for a file format, point to a file using the **Data File(s)** attribute.
   a. Under **Location**, select **Local** to indicate that you want to use a file on the computer where the Data Services Designer is running, or select **Job Server** to choose a file residing on the same computer as the default Job Server you associated with the repository.
   b. When you select **Job Server**, the browse options are disabled, so you must type the absolute path to the File name(s), or you can use a substitution parameter.

      An absolute path on UNIX might look like this:

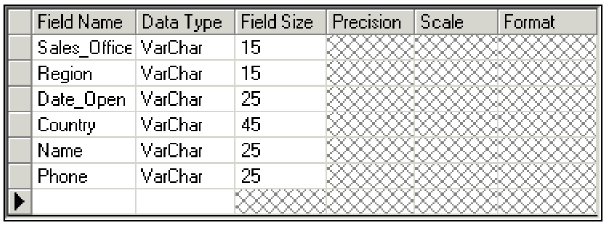
      ```
      /user/data/abc.txt
      ```

      A path on Windows might look like this:

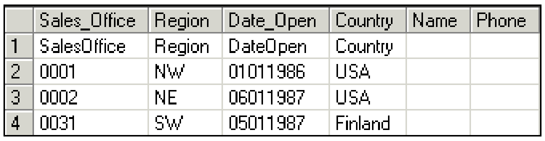
      ```
      C:\DATA\abc.txt
      ```

Note:
The Data File(s) attribute is a design-time tool that allows you to view data and create metadata schema using a data file located on the local or the Job Server computer. When the file format you create is dragged into a data flow as a source or target, you might need to change the File name and Root Directory attributes to reflect the data files used in your job. Your data file(s) must be located relative to the Job Server computer on which your job will be executed.

When you specify the file, the editor populates the Column Attributes work area for editing and places sample data from the file in the Data View work area. As you edit column attributes, the Data View work area changes to reflect those modifications.



Field Name	Data Type	Field Size	Precision	Scale	Format
Sales_Office	VarChar	15			
Region	VarChar	15			
Date_Open	VarChar	25			
Country	VarChar	45			
Name	VarChar	25			
Phone	VarChar	25			



	Sales_Office	Region	Date_Open	Country	Name	Phone
1	SalesOffice	Region	DateOpen	Country		
2	0001	NW	01011986	USA		
3	0002	NE	06011987	USA		
4	0031	SW	05011987	Finland		

Click any cell to edit it. Use mouse or keyboard navigation (tab, arrow keys) to move through the fields and view lists from which you can choose data. Right-click to view a pop-up menu to insert or delete new rows.

7. (Optional) Modify the file format delimiter.
   a. Under **Delimiters** click the **Column** box.
   b. Click the down arrow to view menu options.
   c. Click to choose a delimiter type (**Comma**, **Semicolon**, **Space**, or **Tab**). Alternately, enter an ASCII character. Valid ASCII characters range from /0 to /254.
   d. If defining a file format for a source, verify that the column delimiter matches the file from which you are reading. If defining a file format for a target, you can keep the default value.

   Note:
   The data in the file cannot contain the character(s) used as delimiters.

8. (Optional) To examine the ATL file associated with this file format, click **Show ATL** to view a read-only version of the ATL file.

   To search for specific words or phrases within the file, click **Find** and enter a key word or phrase, then click **Find Next** to find occurrences of the word or phrase. Click **Cancel** when you are finished finding words or phrases. Click **Cancel** again to return to the file format editor.
9. After you edit your column attributes, click **Save & Close**.
10. If you are using an existing file as a model, verify that the fields are delimited and have the data type you expect.

Related Topics

- File format
- Designer Guide: File format

4.8 Real-time jobs

Data Services supports real-time data transformation including receiving messages from ERP systems (such as SAP ERP) or XML-based, analytic applications. "Real-time" means that Data Services reacts to messages as they are sent, performing predefined operations and responding appropriately.

You define the operations performed to process messages and create responses by building real-time jobs using the Data Services Designer and then by converting those jobs to on-demand services using the Data Services Administrator.

This section discusses real-time jobs in the SAP applications environment.

Related Topics

- Designer Guide: Real-time jobs

4.8.1 SAP application table sources in real-time jobs

You can use SAP tables as sources in real-time jobs after you import metadata for the table into your repository. When you drag an SAP table to a data flow definition, it becomes a source.

When the data flow performs a query against the SAP table, it executes an SAP function call to extract the data through the SAP application server.

If you include an SAP table in a join with a real-time message source, Data Services includes the data set from the real-time message source as the outer loop of the join. If more than one supplementary source is included in the join, you can control which table is included in the next outermost loop of the join using the join ranks for the tables.

4.8.1.1 To use an SAP application table in a data flow

1. Import the metadata for the table using a datastore.

   The table appears in the table category under the datastore in the object library.

2. Drag the table into the data flow.
3. Connect a query to the output of the table.

   Because SAP application tables are designed to extract a limited amount of data, the first step after the source would most likely be a query with a WHERE clause that limits the data extracted.
4. In the WHERE clause of the query, add an expression to reduce the amount of data extracted from SAP by this source.

   The join expression in the ON clause of the query includes specific data from the message that the data flow is processing. For example, if the data flow is using the ERP table VBUK to supply delivery status values to respond to an e-commerce application, the join expression would restrict the values from VBUK to the specific sales order referred to in the incoming message. In this case, the join expression restricts the values to sales order number and line item numbers matching the input.

4.8.2 Caching source data

All tables or files used as sources allow you to set a cache option. The option indicates that the data extracted is stored in memory until the data flow processing is complete.

In real-time jobs, you should not cache sources unless the data being cached is small and is unlikely to be updated in the life of the real-time process.

In batch jobs, caching can improve the performance of data flow processing by reducing the number of times a set of data is read from the database or file source. In real-time jobs, however, the improvement in performance provided by caching is minimized by the likelihood that the job reads only a small amount of data from the source for any given message. In addition, because the job reloads cached data only when the Access Server shuts it down and restarts it, cached data may become stale in memory.

4.8.3 Data cache targets and transactional loading

In addition to IDocs, you can use tables and files as targets in real-time jobs. Target tables in data flows also support transactional loading, in which the data resulting from the processing of a single message can be loaded into multiple tables as a single transaction. No part of the transaction is applied if any part fails.

You can specify the order that tables are included in the transaction through an option in each target table editor.

Loading targets as a single transaction only applies to targets in a single datastore. If the real-time job loads tables in more than one datastore, targets in each datastore are loaded independently.

In addition, the options to bulk load a table or to include pre-load or post-load commands are not available for tables used as targets in transactional loading.

Related Topics

- Reference Guide: Objects

Connecting to SAP NetWeaver Business Warehouse

To use an SAP data source as a Data Services source, you must define a Data Services datastore to serve as the logical link to SAP, then import your metadata into the Data Services datastore.

Related Topics

• Designer Guide: Datastores

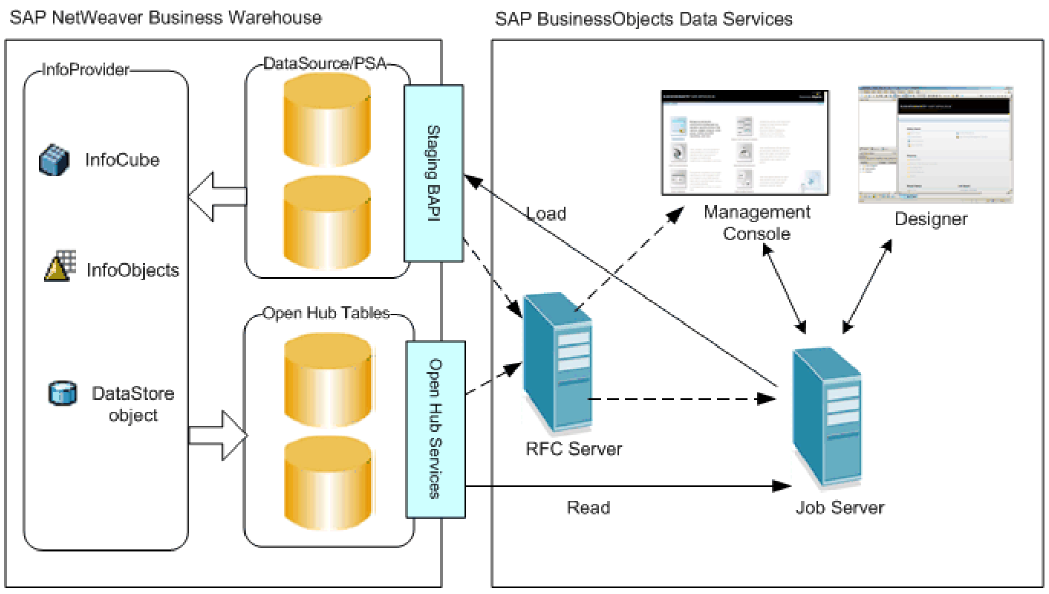
5.1 Using Data Services in SAP NetWeaver Business Warehouse environments

SAP BusinessObjects Data Services provides the following capabilities in an SAP NetWeaver Business Warehouse (BW) environment:

- Read data from an InfoProvider (such as an InfoCube, DataStore object, and InfoObject) in one of the following ways:
  - SAP Open Hub Destination service which executes an SAP NetWeaver BW process chain that reads the data from the InfoProvider and loads an Open Hub Destination table that Data Services reads as a source table.
  - Auto-generated ABAP program or RFC functions that read data from the underlying tables of the InfoCube, DataStore object, or InfoObject.

  The preferred method is to use the SAP Open Hub Destination service to read from a BW InfoProvider.
- Load data into an SAP NetWeaver BW InfoSource by using a Data Services batch job which you can start in one of the following ways:
  - Execute a Data Services batch job that calls a BW InfoPackage automatically to load a BW InfoProvider.
  - Use an InfoPackage in SAP NetWeaver BW to schedule and execute the Data Services load job which has been exported to SAP NetWeaver BW.

The following diagram shows a high-level overview of the components involved with reading data from and loading data to an SAP NetWeaver BW system.



This diagram shows that:

1. Data Services can load data into a DataSource in the Persistent Storage Area (PSA) using the Staging Business Application Programming Interface (BAPI), with support from the RFC Server.
2. Data Services can read data from a BW system by using the Open Hub Service with support from the RFC Server.

Related Topics

- Reading from SAP NetWeaver BW
- Loading into SAP NetWeaver BW

5.2 SAP NetWeaver BW datastores

SAP NetWeaver Business Warehouse (SAP NetWeaver BW) is a business warehouse application that runs on an SAP application server.

You can extract data from the SAP source, transform it as specified in a Data Services batch job, and load it into various targets including a predefined SAP BW InfoSource.

Data Services provides two datastore types for SAP BW: one for sources and one for targets. For other datastore types, you create a datastore, import objects such as tables, then create sources and targets using those objects. For SAP BW, you create separate source and target datastores because they expose different objects.

5.2.1 SAP NetWeaver BW as a source

SAP NetWeaver BW can be used as a source in Data Services by using a SAP BW Source datastore. You can import the same objects as in SAP Applications datastores except hierarchies. In addition, you can browse and import InfoAreas, InfoCubes, ODS objects, and Open Hub tables.

To use SAP NetWeaver BW data as a source:

- Define an SAP BW Source datastore that will serve as the logical link to your BW system.
- Import the SAP BW metadata into Data Services using the SAP BW Source datastore.

5.2.1.1 Defining an SAP BW Source datastore

To define an SAP BW Source datastore:

1. In the Designer's object library, select the **Datastores** tab.
2. Right-click in a blank area of the object library window and select **New**.

   The "Create New Datastore" window opens.
3. Enter a name for the datastore in the **Datastore name** box.

   The name can contain any alphabetic or numeric characters or underscores (_). It cannot contain spaces.
4. For the **Datastore type** box, select **SAP BW Source**.
5. In the **Application server** box, type the host name.

   The host name can be either the host name or an SAP routing string plus the host name. An example of valid syntax for an SAP routing string is:

   ```
   /H/IP_Address_of_local_SAP_router/H/IP_Address_of_target_SAP_router/H/IP_Address_of_target_application_serv
   er
   ```
6. Type the **User name** and **Password**.
7. Click **Advanced** to enter properties for the SAP application server on which BW is running.

   The recommended method to extract data from a BW system is through the Open Hub Destination service. However, it is still possible to use an ABAP program or RFC functions to extract data from the underlying tables of the InfoAreas, InfoCubes, and DataStore objects.

   a. The following Advanced options are always required regardless of which method you use to extract data from a BW system. Ensure that the default values for these Advanced options are what you want:

      - **Language**
      - **Code page**

    - **Client number**
    - **System number**
  b. If you want to use an ABAP program or RFC functions, you must also specify **Data transfer method** and the associated options for each transfer method (for details, see the link in Related Topics below).

8. Click **OK** to add the SAP BW Source datastore.

   The object library displays the newly created SAP BW Source datastore.

You can now browse and import the Open Hub tables or the BW InfoProvider tables that will serve as your SAP BW source.

Related Topics

- Datastore

5.2.1.2 Using the sapnwrfc.ini file

To change or add parameters without modifying the datastore settings, you can specify them in an sapnwrfc.ini file.

Place your sapnwrfc.ini file in the same directory as the RFC client/server program (for Data Services, commonly %LINK_DIR\bin). Or you can specify the DS_NWRFC_INI environment variable in the operating system. For example, you could can place the sapnwrfc.ini file in C:\Temp and specify `DS_NWRFC_INI=C:\Temp`.

You can specify an sapnwrfc.ini file for two different applications in Data Services:

- To use an sapnwrfc.ini file to override datastore settings, edit the SAP datastore in the Designer. From the object library, open the editor for the SAP datastore to configure and click **Advanced**. Under the "SAP" option category, for **Use sapnwrfc.ini**, select **Yes** and enter the appropriate **Destination name**.
- To use an sapnwrfc.ini file for real-time IDocs, configure the RFC client in the Administrator. Expand the **Real-Time** node, and for the Access Server to configure, click **Client Interfaces**. Open an instance of the RFC client, and select the check box to **Use sapnwrfc.ini** and enter a destination name.

Note:
Data Services ignores the parameters USER, PASSWD, and CLIENT in sapnwrfc.ini because storing these security-related parameters could introduce a security risk.

For more information about how to use the sapnwrfc.ini file, refer to the SAP documentation at http://help.sap.com/saphelp_nwpi711/helpdata/en/48/ce50e418d3424be10000000a421937/frameset.htm.

5.2.1.3 Browsing and importing metadata for an SAP BW source

You can import metadata from the object library in several ways:

- By name when you know the name of the object you want to import
- By browsing when you do not know the name of the objects
- By searching when you know the partial name of the object and you have a large number of objects

5.2.1.3.1 To import SAP NetWeaver BW source metadata by name

1. In the object library, select the name of your SAP NetWeaver BW Source datastore.
2. Right-click and choose **Import By Name**.
3. In the Import by Name window, select the type of metadata that you want to import from the Type drop-down list.
   - If you select Open Hub Table, type the name of the table in the Name field.
   - If you select InfoArea, InfoCube or ODS object, the Name field offers a list of available objects. Select the name of the object you want to import.
4. Click **OK**.

   The specified source appears in the object library.

5.2.1.3.2 To import SAP NetWeaver BW source metadata by browsing

1. In the object library, right-click the name of the BW datastore.
2. Select **Open**.

   The Repository Metadata opens in the workspace.
3. Select **External Metadata** to view the BW objects that you can import.

   The BW external metadata browser opens with the Open Hub Tables node expanded by default. You can scroll down to an InfoArea, InfoCube, or DataStore object, expand the object, and browse its lower level content to identify the objects that provide data to it.
4. Select the structures to import.

   For Open Hub Tables, you can select:
   - A parent category (imports all Open Hub Tables)
   - A single Open Hub Table
   - A subset of Open Hub Tables by using the CTRL or CTRL + SHIFT keys

   For InfoAreas, InfoCubes, or DSOs, you can select:
   - A parent category (imports the parent and all its children)
   - A single category
   - A subset of objects by using the CTRL or CTRL + SHIFT keys
5. Right-click and choose **Import**.

The imported structures appear in the object library. You can also open the datastore to see the contents of the datastore in the workspace by using the Repository Metadata option.

5.2.1.3.3 To import SAP NetWeaver BW source metadata by searching

1. In the object library, select the name of your SAP NetWeaver BW Source datastore.
2. Right-click and choose **Search**.
3. Use the Search dialog to find and import metadata.

5.2.2 SAP NetWeaver BW as a target

Data Services can use SAP NetWeaver BW as a target by using an SAP BW Target datastore.

You can execute the job directly from the Data Services Designer or from SAP NetWeaver BW using BAPI functions through the Data Services RFC Server.

5.2.2.1 SAP NetWeaver BW loading profile

If you are using Data Services to load data into SAP NetWeaver BW during production, create an SAP application profile, such as BODS_BW_PROD, that defines additional authorizations for PRODUSER. This profile requires the following authorizations:

- S_RS_ADMWB
- S_RFC

Related Topics

- S_RS_ADMWB
- S_RFC

5.2.2.2 To use an SAP NetWeaver BW InfoSource as a Data Services target

1. Define a Data Services SAP BW Target datastore that will serve as the logical link to your SAP system.
2. Import your SAP NetWeaver BW InfoSource metadata into the Data Services SAP BW datastore.

Related Topics

- Defining an SAP BW Target datastore
- Browsing and importing metadata for an SAP BW target

5.2.2.3 Defining an SAP BW Target datastore

To define an SAP BW Target datastore:

1. In the Designer's object library, select the **Datastores** tab.
2. Right-click in a blank area of the object library window and select **New**.

   The Datastore Editor window opens.
3. Enter a name for the datastore in the **Datastore name** box.

   The name can contain alphanumeric characters or underscores (_). It cannot contain spaces.
4. For the **Datastore type** box, select **SAP BW Target**.
5. In the **Application server** box, type the SAP host name.
6. Type the **User name** and **Password**.
7. Click **Advanced** to verify the following required properties for the SAP application server on which BW is running:
   - **Language**
   - **Code page**
   - **Client number**
   - **System number**
   - **Routing string**

   Include routing string information if your application server is a remote system accessed through a gateway. No SAP application routing permissions are required, but you must use the specified syntax in the **Routing string** field to ensure connection. An example of valid syntax for an SAP routing string is:

   ```
   /H/IP_Address_of_local_SAP_router/H/IP_Address_of_target_SAP_router/H/IP_Address_of_target_application_serv
   er
   ```
8. Click **OK** to add the datastore.

   SAP BW Target datastores display in the object library.

You can now browse and import the metadata for the Master and Transaction Transfer Structures that will serve as your SAP BW target.

For more information, see “Internationalization” in the SAP Library.

Related Topics

- Datastore
- SAP Library - Internationalization

5.2.2.4 Using the sapnwrfc.ini file

To change or add parameters without modifying the datastore settings, you can specify them in an sapnwrfc.ini file.

Place your sapnwrfc.ini file in the same directory as the RFC client/server program (for Data Services, commonly %LINK_DIR\bin). Or you can specify the DS_NWRFC_INI environment variable in the operating system. For example, you could can place the sapnwrfc.ini file in C:\Temp and specify `DS_NWRFC_INI=C:\Temp`.

You can specify an sapnwrfc.ini file for two different applications in Data Services:

- To use an sapnwrfc.ini file to override datastore settings, edit the SAP datastore in the Designer. From the object library, open the editor for the SAP datastore to configure and click **Advanced**. Under the "SAP" option category, for **Use sapnwrfc.ini**, select **Yes** and enter the appropriate **Destination name**.
- To use an sapnwrfc.ini file for real-time IDocs, configure the RFC client in the Administrator. Expand the **Real-Time** node, and for the Access Server to configure, click **Client Interfaces**. Open an instance of the RFC client, and select the check box to **Use sapnwrfc.ini** and enter a destination name.

Note:
Data Services ignores the parameters USER, PASSWD, and CLIENT in sapnwrfc.ini because storing these security-related parameters could introduce a security risk.

For more information about how to use the sapnwrfc.ini file, refer to the SAP documentation at http://help.sap.com/saphelp_nwpi711/helpdata/en/48/ce50e418d3424be10000000a421937/frameset.htm.

5.2.2.5 Browsing and importing metadata for an SAP BW target

To access InfoSource metadata in Data Services, import its metadata into the object library.

You can import InfoSource metadata in one of two ways:

- By name
- By browsing

5.2.2.5.1 To import InfoSource metadata by name

1. In the object library, select the name of the SAP BW Target datastore.
2. Right-click and choose **Import By Name**.
3. In the Import by Name window, specify the Source System and InfoSource names.

Also, specify whether the InfoSource type is Master or Transaction. If the InfoSource type is Master, specify whether the subtype is Text, Attributes, or Hierarchy.

4. Click **OK**.

   The imported objects appear in the object library under the target datastore node.

5.2.2.5.2 To import InfoSource metadata by browsing

1. In the object library, right-click an SAP BW datastore.
2. Select **Open**.

   The metadata browser opens in the workspace.

   From this viewer you can browse both Master and Transaction InfoSources. Both are sorted in alphabetical order.

   Master Transfer Structures contain dimension data, and Transaction Transfer Structures contain fact data.

   A single InfoSource can be loaded from multiple Source Systems. If you expand an InfoSource you can see all the Source Systems that provide data to it.

3. Select the structures to import.
4. Right-click and choose **Import**.

   The imported structures appear in the object library.

5.3 Reading from SAP NetWeaver BW

SAP NetWeaver Business Warehouse provides an Open Hub Service through which SAP BusinessObjects Data Services can read data from an InfoProvider. The Open Hub Service extracts data from an InfoProvider and loads it into an Open Hub Destination object that Data Services reads. Data Services uses the Destination Type Third Party Tool.

The following is an overview of the tasks that you must do to read from an Open Hub Destination table:

- In the SAP Data Warehousing Workbench, define an RFC destination and set up the Open Hub Destination object.
- In the Data ServicesManagement Console, set up the RFC server in the Administrator.
- In the Data ServicesDesigner, do the following:
  - Define an SAP BW Source datastore to connect to the BW System that has the Open Hub Destination table.
  - Import the Open Hub Destination table.
  - Create an Data Services job to start the BW process chain that loads data from the InfoProviders into an Open Hub Destination table, receive notification when the data is available, read the data, and delete the data from the Open Hub Destination table after successfully reading the data.

Related Topics
- Setting up the Open Hub Destination object
- Setting up the RFC server to read an Open Hub table
- Defining an SAP BW Source datastore
- Importing Open Hub tables
- Creating a job to read an Open Hub table
- Running the job to read an Open Hub table

5.3.1 Setting up the RFC server to read an Open Hub table

The Remote Function Call (RFC) Server is required when a job reads from an Open Hub Destination table in SAP NetWeaver Business Warehouse (BW). When you execute the SAP BusinessObjects Data Services job, Data Services first launches the BW process chain to load the Open Hub Table. The RFC Server notifies Data Services when the load has completed. The Data Services job can then read data from the Open Hub Table, transform it, and load the data to a target.

To set up the RFC Server to read Open Hub Destination tables, you must do the following tasks:

- In the SAP Data Warehousing Workbench, define an RFC Destination:
  - Specify a name in **RFC Destination**.
  - Ensure that **Connection Type** has the value TCP/IP Connection.
  - On the **Technical Settings** tab, in the "Registered Server Program" area, specify a name for your RFC Server in **Program ID**.

    The value in **Program ID** is the value that you specify in option **RFC Program ID** when you configure the RFC Server in Data Services.

- In the Management Console of Data Services, go to the Administrator and do the following to configure the RFC Server:
  - Select **Management** > **Repositories** on the navigation tree, and add the repository where you imported the Open Hub table.
  - Select **SAP Connections** > **RFC Server Interface** on the navigation tree, and configure the RFC Server associated with the Open Hub table.

    In the option **RFC Program ID**, specify the value of option **Program ID** when you defined the RFC Server in the Data Warehousing Workbench. The value of attribute "3RDPARTYSYSTEM" for the Open Hub table is also the value of **Program ID**.

  - Start the RFC Server.

    Note:
    You cannot have multiple RFC server instances that refer to the same Program ID running in the Data Services environment.

Related Topics
• Management Console Guide: Administrator, Adding an RFC server interface

5.3.2 Setting up the Open Hub Destination object

Use the SAPData Warehousing Workbench to define an RFC destination and set up an Open Hub Destination object.

1. Define an RFC Destination that points to Data Services RFC Server registered Program ID.
   - Specify a name in **RFC Destination**.
   - Ensure that **Connection Type** has the value TCP/IP Connection.
   - On the **Technical Settings** tab, in the "Registered Server Program" area, specify a name for your RFC Server in **Program ID**.

     The value in **Program ID** is the value that you specify in the option **RFC Program ID** when you configure the RFC Server in the Data ServicesAdministrator on the Management Console.

2. Create an Open Hub Destination and specify the following options:
   - For **Destination Type**, select **Third Party Tool** in the drop-down list.
   - For **RFC Destination**, specify the name that you defined in step 1.
3. Create a Data Transfer Process (DTP) to extract data from an InfoProvider and load it to the Open Hub Destination table.
4. Create a process chain to execute the DTP.

   Note:
   Each process chain can only contain one DTP of the Open Hub Destination.

5.3.3 Importing Open Hub tables

The Open Hub Destination tables must already be defined in SAP NetWeaver BW before you can import their metadata into SAP BusinessObjects Data Services. When you browse the list of Open Hub tables in the SAP BW Source datastore in Data Services, only the Open Hub objects with destination type Third Party Notification are visible.

Note:
Although import by browsing might be the simpler way to import Open Hub tables, the following procedure describes how to import by searching to provide an example if you have a BW system with a large number of objects and browsing might be more difficult.

1. In the object library, select the name of your SAP NetWeaver BW Source datastore.
2. Right-click and choose **Search**.

3. In the "Search" window, do the following actions to find and import metadata.
   a. Ensure that the value for "Select location" is **External Data**.
   b. In the "Object Type" drop-down list, select **Open Hub Tables**.
   c. Choose the "Select criteria" you want and type in the string you want to find.
   d. Click **Search**.
   e. From the search result list, select the table names that you want, right-click, and choose **Import**.

The "Local Object Library" shows the imported objects under the Open Hub Tables node under the datastore name.

Related Topics
- To import SAP NetWeaver BW source metadata by name
- To import SAP NetWeaver BW source metadata by browsing
- To import SAP NetWeaver BW source metadata by searching

5.3.4 Creating a job to read an Open Hub table

This topic describes the usage scenario of one user reading from one Open Hub table.

Note:
When you import Open Hub Tables, you can use them in regular data flows. If you import InfoAreas, InfoCubes, or DataStore objects, Data Services imports the underlying ABAP tables which you use in ABAP data flows so that you can push down the ABAP functions. If you use the ABAP tables in regular data flows, you cannot push down the ABAP functions.

1. In the SAP BusinessObjects Data Services Designer, create a job, a work flow, and data flow within the job.
2. Select an Open Hub table from the local object library and drag it into the open data flow.

   The Open Hub table becomes a source object in the data flow.
3. Double-click the Open Hub source object in the data flow to open the editor and specify the following options.
   a. For **Open Hub Table**, leave the default value (name of the current Open Hub table).
   b. Select **Execute process chain before reading** to start the SAP NetWeaver BW process chain that loads the Open Hub table from which the Data Services Open Hub source object reads the data.
   c. For **Process chain name**, ensure that the default value (first process chain name listed in the attributes for the current Open Hub table) is the correct process chain.

   For details about the options on the Open Hub Table editor, see the Related Topic below.
4. Add the transforms you want to perform on the extracted data and add your target to the data flow.
5. If you want to take specific actions for any errors that might occur while accessing and reading the Open Hub table, add a try/catch block and select the exception group SAP BW execution errors (1009).

6. Save the job and data flow.

Before you execute the job, you must set up the RFC server.

Related Topics
- Setting up the RFC server to read an Open Hub table
- Open Hub Table source
- Designer Guide: Work Flows, Defining a try/catch block

5.3.5 Running the job to read an Open Hub table

The SAP BusinessObjects Data Services job initiates the process chain to load the Open Hub Destination table in SAP NetWeaver BW.

The following must be running for job execution to complete successfully:
- The SAP NetWeaver BW application server where the Open Hub Destination tables reside
- The SAP BusinessObjects Data Services RFC Server
- The SAP BusinessObjects Data Services Job Server

1. From the Designer of the SAP BusinessObjects Data Services, open the Project that contains your Open Hub job.
2. Right-click the job name and select **Execute**.

   The Trace log window opens and you can see the job progress messages. A message appears when the process chain completes.

   If an error occurs, the Error log displays the status of the process chain execution. For information about the status codes for the process chain execution, see the Related Topic below.

Related Topics
- sap_openhub_processchain_execute

5.3.6 Reading multiple Open Hub tables

This section describes examples of when you might want to read from multiple Open Hub tables.

5.3.6.1 Creating a data flow for multiple users to read one InfoProvider

In this scenario, two users want to extract data from the same InfoProvider (InfoCube or DataStore object) in SAP NetWeaver BW to transform and load to a target in SAP BusinessObjects Data Services. For example, one user might want to obtain information about materials in Europe and a second user might want the same materials information in North America. Both users can use the same Data Services data flow and specify their specific Open Hub table and process chain with substitution parameters.

1. In the BW Administrative Workbench, define two different Open Hub Destinations that have the same schema. For each Open Hub Destination, define a Data Transfer Process (DTP) that extracts the data with a filter for each geographic area.

   For example, you might define OpenHubEurope and OpenHubNA.

2. In the Designer of Data Services, in your Open Hub datastore in the "Local Object Library", import the Open Hub Destination tables.
3. Define substitution parameters for the Open Hub table name and process chain name. To add and define a substitution parameter:
   a. In the Designer of Data Services, select **Tools** > **Substitution Parameter Configurations**.
   b. In the Substitution Parameter Editor, scroll to the bottom of the parameter list and double-click the blank cell to enable editing.
   c. Type the name of your substitution parameter, prefaced with $$.

      For example, you can enter $$OpenHubTable.

4. You can define default values in a substitution parameter configuration and associate these defaults with a system configuration.

   For example, you might have an existing system configuration for Europe and one for North America.

   a. Create a substitution parameter configuration for each system configuration: Sub_Parm_Con fig_Europe and Sub_Parm_Config_NA.

      For details about creating a substitution parameter configuration and associating it with a system configuration, see the Related Topic below.

   b. To define default values, double-click in the blank cell next to the substitution parameter name in the column that identifies each substitution parameter configuration and type the constant value.

      For example, you might type OpenHubEurope as the default value for the substitution parameter "$$OpenHubTable" in the column for "Sub_Parm_Config_Europe". Similarly, you might type OpenHubNA in the column for "Sub_Parm_Config_NA".

5. Create a Data Services data flow.
6. Select an Open Hub table from the "Local Object Library" and drag it into the open data flow.

   The Open Hub table becomes a source object in the data flow.

7. Double-click the Open Hub source object in the data flow to open the editor and specify the following options.

   a. For **Open Hub Table**, click the arrow to display the drop-down list, and select the substitution parameter that you defined for the name of the Open Hub Table in step 2.
   b. Select **Execute process chain before reading** to start the SAP NetWeaver BW process chain that loads the Open Hub table from which the Data Services Open Hub source object reads the data.
   c. For **Process chain name**, click the arrow to display the drop-down list, and select the substitution parameter that you defined for the process chain name in step 2.

   For details about the options on the Open Hub Table editor, see the Related Topic below.

8. Add the transforms you want to perform on the extracted data and add your target to the data flow.
9. If you want to take specific actions for any errors that might occur while accessing and reading the Open Hub table, add a try/catch block and select the exception group SAP BW execution errors (1009). For details about how to define a try/catch block, see the Related Topic below.
10. Save the data flow.
11. Before you execute the job, ensure that the RFC server has been set up on the Management Console Administrator:
    - Configure the repository where you imported the Open Hub table.
    - Define the RFC Server associated with the Open Hub table.
    - Start the RFC Server.
12. When you execute the job, select the system configuration associated with your substitution parameter configuration.
    - In the Management Console Administrator, you can select the system configuration on the following pages:
      - Execute batch job.
      - Schedule batch job.
      - Export execution command.

      To override the default value of a substitution parameter, use the **Add Overridden Parameter** option on these pages.
    - In the Data Services Designer, you can select the system configuration on the "Execution Options" tab of the "Execution Properties" window. To override the default value of a substitution parameter, use the **Pick a Substitution Parameter** option on the "Substitution Parameter" tab.

Related Topics

• Designer Guide: Variables and Parameters, Associating a substitution parameter with a system configuration
• Open Hub Table source
• Designer Guide: Work Flows, Defining a try/catch block
• Setting up the RFC server to read an Open Hub table

5.3.6.2 Creating a data flow to join multiple Open Hub tables

In this scenario, one user reads two Open Hub tables to join the data from them. The Data Services job will consist of a work flow that contains the following:

- A script that contains an SAP function to launch the process chains that extract data from the InfoProvider objects and load it to tables in the Open Hub Destination.
- A data flow to read the Open Hub tables.
- A script to send the Open Hub read status to the BW system.

1. In the BW Administrative Workbench, define Process Chains, each with one Data Transfer Process that extracts the data from the InfoProvider and loads to the Open Hub Destination table.
2. In the Data Services Designer, import the Open Hub Destination tables that you want to join.
3. In the Data Services Designer, create a job, create a work flow, and add a script to the work flow.
4. Open the script and add the following:
   a. For each Open Hub table, add the function sap_openhub_processchain_execute to start the Process chain.

      The following sample function specifies the open hub table Materials and the process chain Materials_PC.

      ```
      sap_openhub_processchain_execute('open_hub_datastore','Materials',
      'Materials_PC',$pclogid,$returntxt)
      ```

      For details about this function, see the Related Topic.

   b. If you want to take specific actions for any errors that might occur while reading the Open Hub table, check the return value of the function sap_openhub_processchain_execute.

      For example, the following script commands check the return value, generate an exception, and print the error if the function is not successful.

      ```
      $status = sap_openhub_processchain_execute('open_hub_datastore',
      'Materials','Materials_PC',$lpcogid,
      $returntxt);
      If ($status != 'S') raise_exception ('Materials_PC process chain execution failed ' || $returntxt);
      ```

   c. If you want to take further actions for any errors that might occur while reading the Open Hub table, add a try/catch block around the data flow and select the exception group SAP BW execution errors (1009).
5. Create a data flow and, from the Local Object Library, drag in each Open Hub table that you want to join.
6. For this scenario, you do not need to open the Open Hub Table editor because you can leave the defaults for the following Open Hub source options:
   a. For **Open Hub Table**, leave the default value (the name of the current Open Hub table).
   b. **Execute process chain before reading** is unselected by default. Keep this default because the first script starts the SAP NetWeaver BW process chain that loads the Open Hub table from which the Data Services Open Hub source object reads the data.
   c. **Process chain name** will be disabled because **Execute process chain before reading** is not selected.

   For details about the options on the Open Hub Table editor, see the Related Topic below.
7. Add the Query transform to join the Open Hub tables, add any other transforms you want to perform on the extracted data, and add your target to the data flow.
8. Add a second script after the data flow in the work flow.

9. Open the second script and for each Open Hub table, add the function sap_openhub_set_read_status to the RFC Server to send the status of the Open Hub table read to the BW system.

   For example, the following function specifies the open hub destination Materials.

   ```
   sap_openhub_set_read_status('open_hub_datastore','Materials',
   'X',$returntxt)
   ```

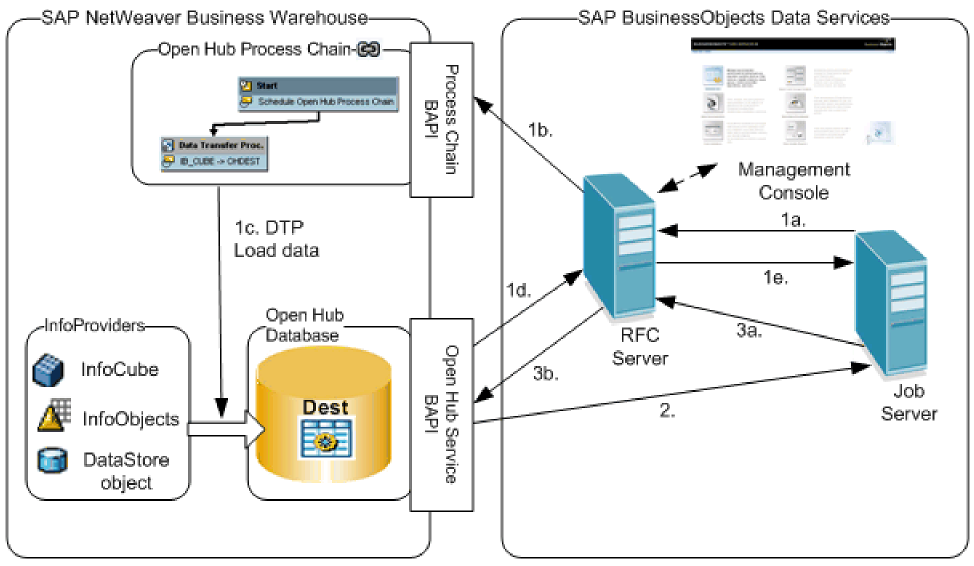
   The function sap_openhub_set_read_status deletes the data from the BW Open Hub Destination.
10. Save the job and data flow.
11. Before you execute the job, make sure you do the following:
    - Configure the repository where you imported the Open Hub table
    - Define the RFC Server associated with the Open Hub table
    - Start the RFC Server

Related Topics

- sap_openhub_processchain_execute
- sap_openhub_set_read_status
- Designer Guide: Work Flows, Defining a try/catch block
- Setting up the RFC server to read an Open Hub table

5.3.6.3 Data Services Open Hub Reader run-time architecture

The following diagram shows a high-level overview of the components involved when using the SAP Open Hub Destination service to read data from an SAP NetWeaver BW system.



This diagram shows:

1. The Data Services job executes function sap_openhub_processchain_execute which starts the following process:

   a. Sends a web service request to the Open Hub RFC Server to start the process chain.

   b. The RFC Server starts the Process Chain in BW system.

   c. The Process Chain contains a Data Transfer Process (DTP) that extracts data from the InfoProvider and loads the Open Hub Destination table.

   d. When the DTP completes, the Open Hub Service notifies the RFC Server.

   e. The RFC Server notifies the Job Server that the Open Hub table is loaded and returns the read request ID of the Open Hub Destination.
2. The Data Services data flow reads the data from the Open Hub table using read request ID.
3. The Data Services job contains a second script that executes the function sap_openhub_set_read_status which starts the following process:

   a. Sends a web service request to the Open Hub RFC Server to set the read status.

   b. RFC Server sets the read status in the Open Hub BW system which deletes the data from the Open Hub Destination table.

5.3.7 Open Hub table restrictions

The following restrictions exist when using Open Hub tables:
- Only one job at a time can read an Open Hub Destination table.
- A process chain of an Open Hub Destination can contain only one of its Data Transfer Processes (DTPs).
- Each process chain cannot contain DTPs for more than one Open Hub Destinations.

5.4 Loading into SAP NetWeaver BW

This section describes the following tasks:
- Preparing the job
- Setting up SAP NetWeaver BW InfoCubes and InfoSources
- Designating Data Services as a source system
- Job execution

You can execute Data services jobs to load to SAP NetWeaver BW from the Designer or you can schedule them from SAP or Data Services.

Note:
This section assumes you are an experienced SAP NetWeaver BW user and have already followed similar procedures. Your system might be configured differently than in the examples, so treat the steps in the procedures as high-level information. You might need to provide additional details or do the steps in a somewhat different order.

5.4.1 Preparing the job

This topic provides an overview of how to prepare to load data into SAP NetWeaver BW.

First, in SAP NetWeaver BW define the appropriate SAP security profiles and authorizations.

In the Data Services Designer:

1. Add datastores for:
   - The system where your source data resides.
   - The system where your SAP NetWeaver BW InfoSource (Transfer Structure) resides (using an SAP BW Target datastore).
2. Import the source and target metadata.
3. Construct and validate the job that will extract data from the source to the SAP NetWeaver BW InfoSource.

In the Data Services Administrator:

1. Add the repository that includes the job.
2. Start the RFC server. In the Administrator, add an RFC Server Interface and start it from the status page.

Related Topics
- SAP NetWeaver BW as a target
- Management Console Guide: Administrator, Starting or stopping an RFC server interface connection
- Starting the RFC server
- Executing the job

5.4.2 Setting up SAP NetWeaver BW InfoCubes and InfoSources

Use the SAP Data Warehousing Workbench to set up SAP BW.

To set up the environment:

1. Create and activate the InfoCube or InfoCubes where the extracted data will ultimately be placed.

   Your InfoCubes should be defined to hold data extracted using Data Services.

2. Create and activate the InfoSource where Data Services will place the extracted data:
    a. Go to the InfoSource window (**InfoSources** tab).
    b. Right-click InfoSources at the top of the hierarchy and select **Create Application Component**.
    c. Complete the window that appears with appropriate information. So for **Application comp**, you might enter: MYAPPCOMP. For **Long description**, you might enter: My application component.
    d. Click **Enter**.

        The application component is created and appears in the hierarchy list.
    e. Right-click the name of your new application component in the component list and select **Create InfoSource**.

        The Create InfoSource: Select Type window appears.
    f. Select the type of InfoSource you are creating (**Transaction data** or **Master data**) and click **Enter**.

        When you select Transaction data, the Create InfoSource (Transaction data) window appears.
    g. Enter the appropriate information and click **Enter**.

        The new InfoSource appears in the hierarchy under the application component name.

5.4.3 Designating Data Services as a source system

If the source system does not exist, right-click and choose **Create source system**, then create the new source system as explained in the following steps:

1. From the SAP Data Warehousing Workbench window, right-click **Source systems** at the top of the hierarchy, and choose **Create source system**.

    The Select Source System Type window appears.
2. Select the last check box on the window and click **Enter**.

    The Create Source System window appears.
3. Enter information about your system, then click **Enter**.

    The RFC Destination window appears.
4. Click **Registered Server Program**.
5. Enter the program ID.
6. Click the **Save** icon to complete registration.
7. Exit this menu and return to the Data Warehousing Workbench InfoSource window.
8. If the Source System already exists, right-click the InfoSource and select **Assign Source System**.

    The Transaction data - InfoSource: Assign source system window appears.
9. Enter the information required and click **Enter**.
10. Click the button to the right of the InfoSource name to view the Maintain Communication Structure window.
11. Add or change information as required.

12. Activate the Transfer Structure for this Source System:
    a. Right-click the InfoSource name and select **Update InfoSource Metadata**.
    b. Enter or change information as required, then click **Save**.

    Note:
    If you will be performing both full and delta updates, check Delta update.

13. Map the transfer structure fields to the communication structure fields.

    Note:
    You must successfully complete all previous steps to begin this step.

    a. Right-click the name of the external source system and select **Maintain Transfer Rules**.
    b. Map the fields and add transformation information as required.
    c. Click **Activate**.

5.4.4 Job execution

You can execute the job from the Designer or from SAP NetWeaver BW.

You can schedule the job to execute from SAP NetWeaver BW, which initiates the request that causes Data Services to extract data from the specified sources and make it available to SAP.

The batch job execution process consists of the following steps:

- If you are using the SAP NetWeaver BW scheduler, it executes code that sends the load request to the Data Services RFC server.

  The load request contains the following information:

  - The name of the batch file you exported from Data Services
  - The request ID, request date, request time, and full/delta update flag
- If you are using the SAP NetWeaver BW scheduler, the Data Services RFC server calls the Job Server which then invokes Data Services to extract and load the SAP NetWeaver BW InfoSource.
- Request information is passed to Data Services which executes the job and loads data to SAP NetWeaver BW with the request ID to identify into which transfer structure it is loading.
- Data Services extracts the data.
- Before loading to SAP NetWeaver BW (through the staging BAPI interface), Data Services first checks whether the InfoSource metadata changed since the last data import.

  If the metadata changed, Data Services produces an error message indicating that you must re-import the Transfer Structure metadata into the Data Services repository before proceeding to import the target data into the Transfer Structure.

5.4.4.1 Executing the job

You can execute the job from the Designer or from SAP NetWeaver BW.

To execute the job from the Designer, right-click the job name in the project area and click **Execute.**

To execute the job from SAP NetWeaver BW, first configure the connection in BW as follows.

1. Create a new InfoPackage.
2. On the **3rd Party Selections** tab, refresh the view to test the connection to the Data Services RFC server interface.

   If the connection is successful, the **3rd Party Selections** tab displays four returned parameters.
3. Enter the **Repository** and **Jobname**.

   The JobServer and Advanced parameters are for advanced use; leave them blank.
4. Refresh the display again.

   The tab now displays any global variables that have been configured for the job so you can provide values for them.

Note:
If you have Data Services Designer or the Administrator open when the job runs, the trace log appears automatically and remains open and active while the job is processing just as it does for a job whose execution is initiated from Data Services.

5.4.4.2 Monitoring the job

You can monitor the jobs and view logs in both Data Services and SAP NetWeaver BW. If you launch the job from BW and the Data Services Designer is running, the log opens automatically as it does when you launch jobs from Data Services.

To view the execution logs in Data Services, in the Administrator, under **SAP Connections**, click **RFC Server Interface** and click the name of the interface to view its most recent log. You can also view the previous two logs.

To view the execution logs in SAP NetWeaver BW, access the InfoPackage. In the **Extraction** monitor, click the **Non-SAP System Extraction Log** icon. The log will indicate whether the job completed successfully or if there were execution errors.

Related Topics

• Management Console Guide: Administrator, Viewing RFC server interface logs

Connecting to SAP Master Data Services

SAP BusinessObjects Data Services lets you create a datastore to the SAP Master Data Services application and import tables for use as sources in Data Services jobs. SAP Master Data Services is powered by the SAP HANA platform.

With the SAP Master Data Services interface, you can use the Designer to:

- Browse and import SAP Master Data Services tables.
- Use SAP Master Data Services tables as sources and targets.

6.1 SAP Master Data Services datastores

SAP BusinessObjectsData Services uses datastore connections to link with other applications or databases.

- In a design environment, use a datastore to browse or import metadata that represents external tables and other database objects.
- In a production environment, Data Services uses datastore information to move data between source and target databases and applications.

After defining an SAP Master Data Services datastore in Data Services, you can browse tables that you want to import. You can also import tables by name.

Related Topics

- Defining an SAP Master Data Services datastore
- Browsing and importing metadata

6.1.1 Defining an SAP Master Data Services datastore

With an SAP Master Data Services datastore connection, you can browse SAP Master Data Services application metadata.

6.1.1.1 To define an SAP Master Data Services datastore

1. In the object library of the Designer, go to the **Datastores** tab
2. Right-click inside the object library window and select **New**.
3. In the "Datastore Editor" window, enter a name for this datastore.

   The name can contain alphanumeric characters and underscores. It cannot contain spaces.
4. In the **Datastore type** list, select **SAP Master Data Services**.
5. Enter the connection information.
6. Click **OK**.

   The SAP Master Data Services datastore appears in the object library.

6.1.2 Browsing and importing metadata

After you create an SAP Master Data Services datastore, you can browse and import the metadata from the connected application.

6.1.2.1 To browse and import metadata

1. Right-click the SAP Master Data Services datastore name and select **Open**, or double-click the datastore name.

   The workspace displays the Tables folder.
2. To view all of the tables, expand the Tables folder.
3. To import table data, right-click one or more tables and select **Import**.

You can also import SAP Master Data Services metadata by right-clicking a Master Data Services datastore in the object library and selecting **Import by name**.

Once tables are imported, you can view column names, data types, and business names for use in creating jobs.

6.1.2.2 Table type

SAP Master Data Services applications use the following table types:

- **read:** Can be used as sources.
- **read_write:** Can be used as sources and targets.
- **write_only:** Can be used as targets.

Reference Information

This section contains SAP-specific reference information about SAP objects, functions, and data types.

For more information on all objects, see the *SAP BusinessObjects Data Services Reference Guide*.

7.1 Objects

7.1.1 Common Data Services objects

The following table lists names and descriptions of SAP BusinessObjects Data Services objects that are associated with SAP Applications and SAP NetWeaver Business Warehouse.

Object	Class	Description
Datastore	Single-use	Describes the connection information needed for Data Services to access a database, an SAP Applications system, or an SAP NetWeaver BW system. Cannot be dropped.
File format	Reusable	Indicates how data is arranged in a source or target file.
Job	Reusable	A job is a set of objects that you can execute together.
Source	Single-use	An object from which Data Services reads data in a data flow or an ABAP data flow.

Object	Class	Description
Table	Reusable	Indicates an external DBMS table for which metadata has been imported into Data Services, or the target table into which data is or has been placed. A table is associated with its datastore; it does not exist independently of a datastore connection. A table retrieves or stores data based on the schema of the table definition from which it was created.
Target	Single-use	An object in which Data Services loads extracted and transformed data in a data flow.

Related Topics
- Reference Guide: Objects

7.1.1.1 Datastore

Class

Reusable

Access

In the object library, click the **Datastores** tab.

Description

A datastore provides a connection to a database. Data Services uses the connection defined by the datastore to import descriptions of the database and its tables (metadata). When you specify tables, hierarchies, or other objects as sources or targets in a data flow, Data Services uses the datastore to determine how to read data from or load data to those tables.

In addition, some transforms and functions require a datastore name to qualify the tables they access.

Data Services provides the following datastore types to connect to SAP systems:
- SAP Applications to read data from or load data to an SAP Applications system
- SAP BW Source to read data from an SAP NetWeaver BW system. The options for SAP BW Source datastores are the same as for SAP Applications.

- SAP BW Target to load data to an SAP NetWeaver BW system. The options for SAP BW Target datastores are common to all SAP datastores.

Note that if you delete a datastore from the object library, you must remove references to the datastore from the following locations:

- Source or target tables using this datastore in your data flows.
- The lookup and key_generation functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references.
- ABAP data flows for SAP Applications connections.

SAP datastores have the following common options:

SAP option	Possible values	Description
Main window		
Datastore name	Alphanumeric characters and underscores	The name of the object. This name appears on the object in the object library and in the calls to the object.
Datastore type	SAP Applications SAP BW Source SAP BW Target	Select the type of datastore to which you are connecting.
Application server	Computer name, fully qualified domain name, or IP address	Name of the remote SAP application computer (host) to which the software connects.
User name	Alphanumeric characters and underscores	Enter the name of the account through which the software accesses the SAP application server.
Password	Alphanumeric characters and underscores, or blank	Enter the user's password. **Note:** If you have problems connecting to a system that uses SAP_BASIS version 6.40 and earlier, the system might be expecting an uppercase password. To prevent this issue, install the appropriate kernel patch as described in SAP Note 792850, "Preparing ABAP systems to deal with incompatible passwords."
Locale		

SAP option	Possible values	Description
Language	SAP-supported ISO two-letter language codes or <default>	Select the login language from the possible values in the drop-down list. The <default> option sets the language to what is set in the SAP system user profile.
Code page		Specify the character encoding of character data in the source in the following situations: • For ABAP data flow processing in SAP R/3 4.x systems • For processing IDoc files Also see "Reference Guide: Locales and Multi-byte Functionality".
SAP		
Client number	000-999	The three-digit client number. Defaults to 800.
System number	00-99	The two-digit system number. Defaults to 00.
Routing string	Refer to the requirements of the application	Enter the SAP routing string used to connect to SAP systems through SAProuters.
Use sapnwrfc.ini	Yes, No	Select to use an sapnwrfc.ini file, which overrides the datastore settings. Place the sapnwrfc.ini file in the current directory of the process being executed (%LINK_DIR/bin). Defaults to **No**.
Destination name	Refer to the requirements of the application	If using an sapnwrfc.ini file, enter the destination name to reference.
Load balance		
Load balance	Yes , No	Select **Yes** to enable load balancing.
MS host	Computer name, fully qualified domain name, or IP address	Specify the message server host name. Overrides the setting in sapnwrfc.ini.

SAP option	Possible values	Description
MS service	Refer to the requirements of the application	Specify this parameter only if the message server does not listen on the standard service sapms<SysID> or if this service is not defined in the services file and you need to specify the network port directly. Overrides the setting in sapnwrfc.ini.
Server group	`User input`, PUBLIC, SPACE	Optionally specify the group name of the application servers. Default: PUBLIC. Overrides the setting in sapnwrfc.ini.
System ID	Refer to the requirements of the application	Name of the SAP system. Overrides the setting in sapnwrfc.ini.
Authentication		
Authentication	Password, SNC	**Password**: The default. Deploys the user name and password entered in the datastore main window. **SNC**: Select to enable secure network communications (SNC) security and complete the remaining options in this section.
SNC library	Full file path and name of SNC security library	Enter the full path and name of the third-party security library to use for SNC communication (authentication, encryption, and signatures).Overrides the setting in sapnwrfc.ini. Alternatively you can configure the environment variable SNC_LIB.
SNC name of Data Services	Refer to the requirements of the application	Enter the SNC name that the SAP system uses to identify Data Services. Overrides the setting in sapnwrfc.ini.
SNC name of SAP system	Refer to the requirements of the application	Enter the SNC name of the SAP system for this connection. Overrides the setting in sapnwrfc.ini.

SAP option	Possible values	Description
SNC quality of protection	Max available Authentication Integrity Privacy	**Max available**: Maximum quality of protection that is supported by the target SAP system. This value is configured in the SAP Application Server profile parameter snc/data_protection/max. It could be configured to be Authentication, Integrity, or Privacy. **Authentication**: With this protection level, the system verifies the identity of the communication partner, which in this context is Data Services. This is the minimum protection level offered by SNC. No actual data protection is provided. **Integrity**: With this protection, the system detects any changes or manipulation of the data that might have occurred between the two end points of a communication. **Privacy**: With this protection, the system encrypts the messages being transferred to make eavesdropping ineffective. Privacy protection also includes integrity protection of the data. This is the maximum level of protection provided by SNC. Overrides the setting in sapnwrfc.ini.

SAP Applications

The following table lists the options specific to the SAP Applications datastore. The options change depending on which **Data transfer method** you select.

SAP Applications option	Possible values	Description
SAP		

SAP Applications option	Possible values	Description
ABAP execution option	Generate and Execute Execute Preloaded	Select the job execution strategy. Your choice affects the required authorizations. **Generate and Execute:** The ABAP created by the job resides on the same computer as the Job Server and is submitted to SAP using the SAP RFC_ABAP_IN-STALL_AND_RUN function or /BODS/RFC_ABAP_IN-STALL_AND_RUN function. Select this option if the job changes between scheduled executions. **Execute Preloaded:** ABAP resides on the SAP applica-tion server and is submitted using Data Services RFC function modules. Select this option if the job does not change between scheduled executions.
Execute in background (batch)	Yes, No	Specify whether the generated ABAP programs created by SAP application data flows defined with this datastore will execute in batch mode on the SAP server. Batch mode operation is slower than the normal console mode; however, choose batch mode if the application is too long to run during the console mode time frame. Defaults to No.
Target host	Computer name, fully qualified domain name, or IP address	If you chose to execute ABAP programs in the back-ground, specify the target computer (host).
Job class	A, B, C	If you chose to execute ABAP programs in the back-ground, specify the job class.

SAP Applications option	Possible values	Description
Data transfer method	Direct download RFC Shared directory FTP Custom transfer	Define how to retrieve data from the SAP application server to the Data Services server: **Direct download:** Use the SAP WS_DOWNLOAD function to stage the data to the **Local directory**. The Job Server reads from the file, then deletes it. Direct download is not recommended for production use because it cannot handle large amounts of data. **RFC**: Use to stream data from the source SAP system directly to the Data Services data flow process using RFC. For secure data transfer, configure **SNC** authentication with the required SNC quality of protection (in the datastore **Authentication** options). **Shared directory:** Both SAP BusinessObjects Data Services and SAP have direct access to the directory where data is stored. This method works best when both systems are on Windows and security is not an issue. Use NFS (shares one network drive or directory). **FTP:** Use FTP. **Custom transfer:** Use a third-party program. When you select a data transfer method, the appropriate options for that method appear below the option.
ABAP RFC destination	SAPDS or `Destination name`	For the **RFC** data transfer method, enter a TCP/IP RFC destination. You can keep the default name of SAPDS and create a destination of the same name in the source SAP system, or you can enter a destination name for an existing destination.
Working directory on SAP server	Directory path or click **Browse**	A directory on the SAP application server where the software can write intermediate files. All the files used by the data flow should be placed in this directory. This directory also stores the transport file used by the FTP, shared-directory, and custom-transfer data transfer methods. By default, the value in this field uses the value that was typed into the **Application server** field. For example, if the value sap01 was typed into the Application server field, the value of **Working directory on SAP server** becomes \\sap01\.
Custom transfer	These options are visible if you selected the **Custom transfer** data transfer method.	

SAP Applications option	Possible values	Description
Custom transfer local directory	Directory path or click **Browse**	The client-side directory to which data from the SAP server downloads.
Custom transfer program executable	Refer to the requirements of the application	The name of the third-party file transfer program you want to use to transfer files from the SAP working directory to the local directory.
Custom transfer user name	Refer to the requirements of the application	(optional) Login ID for the SAP application server to which the custom transfer program connects.
Custom transfer password	Refer to the requirements of the application	(optional) Password for the SAP application server to which the custom transfer program connects. Passwords entered into this option are encrypted.
Custom transfer arguments	Refer to the requirements of the application	(optional) Specify arguments for your custom transfer program. Arguments can add security or compression mechanisms to your program or include Data Services system variables.
Application path to the shared directory	Directory path or click **Browse**	If you selected the **Shared directory** data transfer method, indicate the path from the Data Services server to the SAP application server's working directory. By default, the value in this field uses the value that was typed into the **Application server** field. For example, if the value sap01 was typed into the Application server field, the value of **Working directory on SAP server** becomes \\sap01\.
Local directory	Directory path or click **Browse**	If you selected the **Direct download** or **FTP** data transfer method, select a client-side directory to which data from the SAP server downloads.
Generated ABAP directory	Directory path or click **Browse**	Indicate the directory into which the software writes generated ABAP files. It can be the same directory as the local/direct download directory. The path is relative to the Job Server.
Security profile	Refer to the requirements of the application	Specify the security profile you want to use in the generated ABAP program. The user of the SAP datastore must have the required profile.
Routing string	Refer to the requirements of the application	Enter the SAP routing string used to connect to SAP systems through SAProuters.

SAP Applications option	Possible values	Description
Number of connection retries	Positive integer	The number of times Data Services tries to establish a connection with the SAP application server. Defaults to 1.
Interval between retries (sec)	Positive integer	The time delay in seconds between connection retries. Defaults to 10.
Upload attributes		
Status	P - SAP Standard Production Program K - Customer Production Program S - System Program T - Test Program	Indicates whether the program is a test program, a system program, or a production program. Default is **T - Test program**. The parameter can have only the value code or the value code and description, separated by a space.
Application	Refer to the drop-down list for available options	Indicates the application area to which the program belongs (Basis, General Ledger, Sales, and so on). The default value is **S - Basis**. The parameter can have only the value code or the value code and description, separated by a space.
Development class (Package)	Refer to the requirements of the application	Indicates the name under which related objects in the ABAP Workbench are grouped together in a package. Default is $TMP. The program is created as a local (non-transportable) object.
Request ID	Refer to the requirements of the application	Indicates the Change and Transport System (CTS) request ID. The default value is blank. This option is populated by Data Services if a non-local program object is created in SAP.
Task ID	Refer to the requirements of the application	Indicates the CTS task ID. The default value is blank. This option is populated by Data Services if a non-local program object is created in SAP.
FTP	These options are visible if you selected the **FTP** data transfer method.	
FTP relative path to the SAP working directory	Directory path or click **Browse**	Indicate the path from the FTP root directory to the SAP server's working directory. When you select FTP, this directory is required.

SAP Applications option	Possible values	Description
FTP host name	Computer (host) name, fully qualified domain name, or IP address	Must be defined to use FTP.
FTP user name	Alphanumeric characters and underscores	Must be defined to use FTP.
FTP password	Alphanumeric characters and underscores, or blank	Enter the FTP password.

SAP Master Data Services

The following table lists the option specific to the SAP Master Data Services datastore. For more information about all other options, see the SAP HANA database datastore section of the *SAP BusinessObjects Data Services Reference Guide.*

SAP Applications option	Possible values	Description
SAP Master Data Services		
Schema owner	Refer to the SAP Master Data Services documen-tation	Enter the schema owner name for foundation tables (the name is usually apps).

Related Topics

- Authorizations for Data Services
- Data Services system variables to transfer a file from SAP applications
- Using the sapnwrfc.ini file
- Reference Guide: Objects, Database datastores, SAP HANA

7.1.1.2 File format

Class

Reusable

Access

In the object library, click the **Formats** tab.

Description

A file format describes the structure of an ASCII file. A file format can be specific to a single file, or it can be a generic description used for many data files. Construct file formats from the properties found in the file format editor.

The built-in file format for SAP is the Transport_Format which is under the **Flat Files** node. The following options are for the SAP transport type only.

Property	Possible values	Description
General		
Type	SAP transport	The format of the data in the text file. Available properties change based on the selected file format type.
Name	Any alphanumeric character and underscores (_)	A descriptive name for the file format. This name appears in the object library.
Data File(s)		
Location	Local, Job Server	(Optional) The computer on which the data file(s) are stored.
File name(s)	File name, file name including full path name, or blank	In new and edit modes, specify an existing file on which you base the file format description. Data from this file appears in the Column Attributes area. In these modes, you can leave this property blank. In source and target modes, specify the location of the actual file for this source or target. In these modes, you cannot leave this property blank. For added flexibility, you can enter: • A variable that is set to a particular file with full path name. • A list of files, separated by commas, or a file name containing a wild card. In this case, Data Services reads all these files as a single source.
Delimiters		
Column	Tab, Semicolon, Comma, Space	Specify the character that indicates the end of one column and the beginning of the next. Alternatively, enter an ASCII character. Valid ASCII characters range from /0 to /254.

Related Topics
- Reference Guide: Objects, File format

7.1.1.3 Job

Class

Reusable

Access
- In the object library, click the **Jobs** tab.
- Right-click a job in the project area.

Description

A job is a set of objects that you can execute together. For Data Services to execute the steps of any object, the object must be part of a job. A single job must include all the steps you want executed.

The following are the additional references to SAP applications.
- Data Services retrieves results from the following types of steps:
  - ABAP data flows
  - ABAP data files
- Jobs have properties that determine what information Data Services collects and logs when running the job. Use Trace properties to select the information that Data Services monitors and writes to the trace log file during a job. You can turn several SAP application-related traces on and off.

Trace	Description
ABAP Query	Writes a message when an ABAP program is submitted and when the ABAP program completes. Also writes the ABAP job status every 15 seconds.
IDoc file reader	Writes a message when an IDoc file reader starts and when it finishes. This trace also reports: • The name of the file that the reader opens • The file status (succeeded or failed) • Total number of IDocs processed from the file

Trace	Description
RFC Function	Writes messages when Data Services calls an RFC function or BAPI. Specifically, this trace writes messages when: • An RFC function or BAPI starts • Processing for an individual record starts • Processing for an individual record finishes • New connections to SAP applications are established during real-time jobs • An RFC function or BAPI finishes
SAP Table Reader	Writes messages when Data Services reads from an SAP application table source. Specifically, this trace writes messages when: • Data Services establishes a connection with SAP • Data Services begins reading from the SAP application table • The RFC call to retrieve table content starts • The RFC call to retrieve table content finishes • Data Services finishes reading from the SAP application table

Related Topics

• Reference Guide: Objects, Log

7.1.1.4 Source

Class

Single-use

Access

- To insert an SAP application table as a source, open the object library, go to the **Datastores** tab, select the table, drag it into a batch or ABAP data flow in the workspace, and select **Make Source**.
- To insert a file as a source, open the object library, go to the **Formats** tab, select a flat file format for the file, drag it into the workspace, and select **Make Source**. Use the file format editor to specify the file name.

- To insert an Open Hub table as a source in a data flow, select the table from the local object library **Datastores** tab and drag it into the work space.
- To insert an SAP extractor as a source in a data flow or an ABAP data flow, select it from the local object library **Datastores** tab, drag it into the work space, and modify the import options as necessary.
- To insert an ABAP data flow as a source, open the object library, go to the **Data flows** tab, select the ABAP data flow, drag it into a batch or ABAP data flow in the workspace.
- To insert an XML message or file as a source, open the object library, go to the **Formats** tab, select the DTD or XML Schema format for the XML file, drag it into the workspace, and select **Make XML message source** or **Make XML file source**. Use the source editor to specify the test file name for the message or the source file name for the file.
- To view options of tables or files, click the name of the table or file in the workspace or in the project area. This opens the table or file editor.
- To view options of ABAP data flows, right click the data flow in the workspace or project area, select **Properties**, and go to the **Options** tab.
- To insert an IDoc message or file as a source, open the object library, click the **Datastores** tab, click an SAP Applications datastore, click the IDoc category, select an IDoc, drag it into a data flow, and select **Make IDoc message source** or **Make IDoc file source**. Use the source editor to specify object attributes.

  Note:
  A real-time job source can be one and only one XML or IDoc message. If another message source is already in the data flow definition, Data Services will not allow you to place the IDoc in the workspace.

Description

A source is an object from which Data Services reads data.

In a batch job, a source can be a table, an SAP application table, a previously defined template table, a file, an XML file, an IDoc file, ABAP data flow, or an SAP extractor.

In a real-time job, a source can be a table, an SAP application table, a previously defined template table, a file, an IDoc file, an XML file, an IDoc message, or an XML message. Each real-time job must have exactly one real-time data source—either an IDoc message or an XML message.

In an ABAP data flow, a source can be another ABAP data flow, an SAP application table, a file located on the SAP application server, or an SAP extractor. Files must be in a file format that SAP can read. Data Services provides a predefined flat file format called Transport_Format that SAP can read.

You can select an SAP application table as a source in any data flow. However, when you use an SAP application table as a data flow source, Data Services extracts data using the Data Services-provided RFC (remote function call) /BODS/RFC_STREAM_READ_TABLE. If this function is not loaded to the SAP application server, Data Services extracts data using the SAP-supplied function, RFC_READ_TABLE. This function call limits extracted data to 512 bytes per row.

When you use an SAP application table as a source in an ABAP data flow, Data Services extracts data using ABAP. For large amounts of data (larger than a few rows), performance is generally better when you extract data from SAP using an ABAP data flow.

With SAP extractor sources, Data Services automatically detects whether the extractor uses the ODP data replication API, which provides optimal performance and changed-data capture (delta) capability.

In a regular data flow Data Services uses RFC to call the ODP data replication API. In an ABAP data flow, Data Services generates the ABAP that calls the ODP data replication API.

IDocs are used as message sources in real-time jobs and as file sources in batch or real-time jobs. Data flows in real-time jobs that use IDocs as message sources can be processed one at a time or in parallel. IDoc message sources are generated directly from SAP.

You can configure an IDoc file source in a data flow of a batch job to read files in groups and use variable file names.

Note:
IDocs cannot be used as source or target objects in ABAP data flows.

A table source has options in addition to the datastore information:

Option	Description
Join rank	Indicates the rank of the table relative to other tables joined in a data flow. The software joins tables with higher join ranks before it joins tables with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.

Option	Description
Cache	Indicates whether Data Services should read the required data from the table and load it into memory. Because an inner table of a join must be read for each row of an outer table, you may want to cache a table when it is used as an inner table in a join and when the data retrieved will fit into available memory. Best practice is to specify the cache only in the Query transform editor. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. The default option in the Query editor is Automatic. The automatic setting in the Query editor carries forward the setting from the source table. **Note:** If any one input schema has a cache specified in the Query editor, the Data Services Optimizer considers only Query editor cache settings and ignores all source editor cache settings. For non-SAP application tables, there are two options: • **Yes**: The table is always cached unless it is the outer-most table in a join. • **No**: The table is never cached. For SAP application tables in an ABAP data flow, there are three options: • **Yes**: The table is always cached unless it is the outer-most table in a join. • **No**: The table is never cached. • **Automatic**: Data Services determines whether to cache the table. The default option is **No**. When two SAP application tables are involved in an inner join and you cache both tables, Data Services improves performance by generating ABAP using the array operations SELECT FOR ALL ENTRIES and IN_MEMORY MERGE JOIN. These operations are exclusive to inner joins, specifically two-way inner joins. Data Services sorts then merges the inner and outer tables to optimize join performance. To enable these operations, set the cache option for both tables to **Yes**. The result is that both inner and outer tables are read into internal tables prior to the join. Use **Package size** on the outer loop of the join to limit the size of internal tables.
Package size	Indicates the maximum number of rows loaded into memory. Limiting rows reduces memory consumption. Package Size is an extension of the SELECT_INTO_TABLE operation, which uses one SELECT statement to retrieve blocks of rows to create an internal table. Use this option when caching a table that might consume excessive memory. Data Services does not limit rows when you enter 0, a negative number, or when you set **Cache** to **No**.
Array fetch size	Indicates the number of rows to load into the ABAP memory table before Data Services fetches the rows. Avoids exceeding memory capacity on the source side. Default is 1000.

Option	Description
Execute in back-ground (batch)	If the SAP table reader is taking more time to execute than what is in the SAP system dialog work process setting rdisp/max_wprun_time (SAP RZ11 transaction), select this option to avoid exceeding the time limit. Ensure an **RFC destination** has been specified in the datastore editor for this source. If there is no RFC destination defined, create an RFC destination object (SAP transaction sm59) and enter the name in the datastore definition.

Note:
Data Services automatically uses the SELECT_INTO_TABLE operation when you cache a database table and the content of the cache populates an internal table, or when you cache one or more LOOKUP functions against one or more tables and the contents of the LOOKUP cache populates an internal table.

A file source also has the **Join Rank** and **Cache** options in addition to the file format information. Data Services interprets **Join Rank** and **Cache** the same in a file as a table.

An XML message source has these options in addition to read-only XML Schema or DTD format information:

Option	Description
XML test file	The location relative to the Job Server of an XML-formatted file to use as the message source when you execute the job in test mode.
Enable validation	A check box to turn on the comparison of the incoming message to the stored XML Schema or DTD format. When this option is selected, the real-time job throws an exception if the incoming message is not valid. When you are developing a real-time job, this validation helps you to make sure sample data is both valid and well-formed. If you select this option in production, make sure to include appropriate error handling either in the Data Services job or the client application to process an error caused if a data flow in the real-time job receives data that does not validate against the imported format.
implied: Join rank	The XML message source is always the outer table in a join. You cannot change its join rank. This option is implied and does not appear in the editor.

An XML file source has these options in addition to read-only XML Schema or DTD format information:

Option	Description
XML file	The location relative to the Job Server of an XML-formatted file to use as the source.

Option	Description
Enable validation	A check box to turn on the comparison of the incoming data to the stored XML Schema or DTD format. When this option is enabled, the data flow throws an exception if the incoming source is not valid.
implied: Join rank	The XML file source is always the outer table in a join. You cannot change the join rank for it. This option is implied and does not appear in the editor.

An IDoc message source in a real-time job has the following options:

Option	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc test file	Browse the network to find a test file to read.
IDoc Type	(Read-only) Data Services will only process IDoc files (listed in the IDoc Test File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.

An IDoc file source, in a real-time job, has the following options

.

Option	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc File/s	Browse to a network file that will contain IDocs generated in SAP. Data Services reads the contents of this file/s when this data flow runs.
IDoc Type	(Read-only) Data Services will only process IDoc files (listed in the IDoc File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.
Join Rank	Indicates the rank of the table relative to other tables joined in a data flow. The software joins tables with higher join ranks before it joins tables with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.

Option	Description
Include file name column	Determines whether to include a column in the source output that contains the source IDoc file name. Defaults to **No**. Change the value to **Yes** when you want to identify the source IDoc file in situations such as the following: • You specified a wildcard character to read multiple source IDoc files at one time • You load from different source IDoc files on different runs
Modify	If the file name is included, this button enables you to modify **File name column** and **Column size**.
File name column	If the file name is included, the name of the column that holds the source IDoc file name. Defaults to **DI_FILENAME**.
Column size	If the file name is included, the size (in characters) of the column that holds the source file name. Defaults to **100**. If the size of the file name column is not large enough to store the file name, truncation occurs from the left.
Include path	If the file name is included, determines whether to include the full path name of the source IDoc file. Defaults to **No**.

Related Topics
- IDoc sources in batch jobs
- Designer Guide: Embedded Data Flows
- IDoc file

7.1.1.5 Table

Class

Reusable

Access

In the object library, click the **Datastores** tab.

Description

You can use a table from a non-SAP Applications datastore as a source or target in a data flow. You cannot use a table from a non-SAP Applications datastore in an ABAP data flow.

You can use an SAP application table as a source or target in an ABAP data flow or as a source in a data flow. You cannot use an SAP application table as a target in a data flow.

Tables have the following built-in attributes:

Attribute	Description
Name	The name of the object. This name appears on the object in the object library and in the calls to the object.
Description	Your description of the table.
Business_Name	Business name imported with table metadata.
Table_Usage	How the object is used when it was accessed most recently. Expect values such as Source, Target, Lookup Table, or Comparison Table.
Total_Number_Of_ Rows_Processed	The number of rows loaded into the table in the last successful load.
Date_last_loaded	The time the table was last successfully loaded by Data Services.
Number_Of_Rows_ Rejected	The number of rows rejected in the last successful load.
Number_Of_Inserts	The number of rows inserted in the last successful load.
Number_Of_Updates	The number of rows updated in the last successful load.
Date_Created	The date that the object was created.
Estimated_Row_ Count	A configurable estimate of the table size used in calculating the order tables are read to perform join operations. Used for SAP application tables only.

Attribute	Description
Number_Of_Deletes	The number of rows deleted in the last successful load.
Elapsed_Time_For_ Load	The time it took to load this table in the last successful load.
Table_Type	The type of datastore object. Most tables are type Table.
SAP_Table_Class_ Name	The table class name as designated by the SAP_Table_Class value.
SAP_Table_Class	Table type information. Used for SAP application tables only. 1 — Transparent table 2 — INTTAB 3 — Cluster 4 — Pool 5 — View 6 — Append (Structure)

7.1.1.6 Target

Class
Single-use

Access
- To display target options, click the name of the target in the workspace or in the project area. This opens the object editor.
- To display target properties, right click a target and choose **Properties**.

Description

A target is an object in which Data Services loads data.

In a batch job, a target can be a flat file, a table or template table, an XML file, an IDoc file, an IDoc message, or an ABAP data flow.

In a real-time job, a target can be a file, a table, a template table, an XML file, an XML message, an IDoc file, or an IDoc message.

In an ABAP data flow, a target can be a file or a data transport object.

Target files

You can use any flat file format in a batch or ABAP data flow as a target. You can use any DTD or XML Schema format for XML file and XML message targets.

If the schema defined in the format doesn't match the schema that is input to the target, Data Services provides validation errors to identify the mismatch.

Target tables

No tables can be used in ABAP data flows as targets.

InfoSource or DataSource target

The following table contains option and description information specific to SAP NetWeaver BW InfoSource and SAP NetWeaver BW DataSource target tables.

See the *SAP BusinessObjects Data Services Reference Guide* for option information common to all target tables.

Option	Description
Options	

Option	Description
Number of loaders	Loading with one loader is known as "single loader loading." Loading when the number of loaders is greater than one is known as "parallel loading." The default number of loaders is 1. You can specify any number of loaders. When parallel loading, each loader receives the number of rows indicated in the "Rows per commit" option, and applies the rows in parallel with the other loaders. For example, if you choose a **Rows per commit** of 5000 and set the **Number of Loaders** to 3: in parallel, the loaders each receive a maximum of 5000 rows; then each loader performs the necessary processing and, when complete, sends the data in a packet to the SAP NetWeaver BW system. Due to differences in processing, the loaders may not send the packets to the SAP NetWeaver BW system in sequential order. **Note:** Parallel loading is not supported for the hierarchy type.

Target Idocs

IDoc messages can be sent from Data Services on a regular schedule or in real-time. Data Services allows you to use a batch or real-time job to configure IDoc message targets. Any number of IDoc targets can be included in a single data flow.

IDoc targets support reduced message types. When Data Services detects a reduced message type, it retrieves the appropriate metadata from SAP and creates a reduced IDoc, removing unused segments and giving unused fields the value of '/'.

Note:
Reduced message type functionality is only available for some message types (e.g., DEBMAS, CREMAS, etc.).

The attributes of an IDoc target depend on the type of target and the type of data flow. To see the IDoc attributes, double-click an IDoc target object in a data flow diagram.

- IDoc message targets in a data flow of a batch job

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc Type	(Read-only) Displays the IDoc Type you imported Data Services into for this object. For example, SOPGEN01.
Partner Number	This is an SAP term for where to send the IDoc in the SAP application. Enter the partner number of the IDoc you will be processing as a message target in Data Services. If you do not know the partner number, consult your SAP administrator.
Partner Type	This is an SAP term for an attribute of a partner number. Enter the partner type. For example, `LS` for logical system.
Message Type	This is an SAP term for an EDI message for this IDoc Type. Enter the message type.
Batch size	Specifies the number of IDoc messages to accumulate before Data Services sends the IDoc messages to the SAP server. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of seconds is less than the value in **Batch wait timeout**.
Do not create TID	This attribute improves performance by not creating a transaction ID for SAP transmissions. However, transaction services (such as re-transmission and checking) are not done.
Batch wait timeout	This attribute limits the wait time (in seconds) before Data Services sends the IDoc messages to the SAP server. A value of `0` specifies to wait indefinitely. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of messages is less than the value in **Batch size**.

- IDoc message targets in a data flow of a real-time job

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc Test File	Browse to a network file that will contain IDoc/s generated in Data Services. For example: `C:/temp/`*`filename`*. Use this location to test the output of the IDoc message target before starting a real-time job in the Data Services Administrator. Data Services writes to this file when this real-time job is executed in test mode.
Partner Number	This is an SAP term for where to send the IDoc in SAP. Enter the partner number. If you do not know the partner number, consult your SAP administrator.
Partner Type	This is an SAP term for an attribute of a partner number. Enter the partner type. For example, LS for logical system.
Message Type	This is an SAP term for an EDI for this IDoc Type. Enter the message type.
Batch size	Specifies the number of IDoc messages to accumulate before Data Services sends the IDoc messages to the SAP server. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of seconds is less than the value in **Batch wait timeout**.
Do not create TID	This attribute improves performance by not creating a transaction ID for SAP transmissions. However, transaction services (such as re-transmission and checking) are not done.
Batch wait timeout	This attribute limits the wait time (in seconds) before Data Services sends the IDoc messages to the SAP server. A value of `0` specifies to wait indefinitely. This attribute improves performance because Data Services sends multiple IDoc messages with each transmission. Data Services sends the accumulated IDoc messages even if the number of messages is less than the value in **Batch size**.

Attribute	Description
IDoc Type	(Read-only) Data Services will only process IDoc files (listed in the IDoc Test File box) if they are of this IDoc type. For example, SOPGEN01. This information is entered when you import metadata for the IDoc Type.

- IDoc file targets in data flows

Attribute	Description
Make Port	(Optional) Select an IDoc as an input port for an embedded data flow.
IDoc File	Browse to a network file that will contain IDocs generated in Data Services. Data Services writes to this file when the data flow is executed. When you configure an IDoc File target, Data Services allows you to capture data for more than one IDoc. Each time a root segment passes into a file target (for example **IDoc File**`C:\temp\idoc`), a new file is created and named using an eight character suffix starting with 00000000.txt. For example an **IDoc File** set to `C:\temp\idoc` could store idoc00000000.txt to idoc99999999.txt.
IDoc Type	(Read-only) The IDoc type you imported from SAP into Data Services for this object. For example, SOPGEN01.

Related Topics
- Reference Guide: Objects, Target

7.1.2 SAP-specific objects

This section describes SAP-specific objects available in Data Services.

Object	Class	Description
ABAP data flow	Reusable	Extracts data from SAP application sources.

Object	Class	Description
Custom ABAP transform	Reusable	Inserts custom ABAP logic blocks into a generated ABAP program for an ABAP data flow.
Data transport	Single-use	Identifies a file to hold a data set extracted by an ABAP data flow.
Extractor source	Single-use	An extractor is a source object that Data Services uses to read data from an SAP application.
Hierarchy	Reusable	Browse, search, and import hierarchy metadata.
IDoc file	Reusable	An intermediate document that SAP produces to move data between servers. In Data Services, an IDoc file can be used in a real-time or batch job as a source or a target.
IDoc message	Reusable	An intermediate document that SAP produces to move data between servers. In Data Services, an IDoc message can be used in a real-time job as a source or a target and in a batch job as a target.
Open Hub Table source	Single-use	A source object that Data Services uses to read data from an SAP NetWeaver Business Warehouse system.
Transport_Format	Reusable	The Transport_Format is a Data Services predefined file format object for reading flat files in SAP applications. It is automatically used to define data transport objects added to ABAP data flows.

Related Topics
- The Transport_Format
- File format

7.1.2.1 ABAP data flow

Class

Reusable

Access

- To insert an existing ABAP data flow, go to the object library **Data Flows** tab, open the list of ABAP data flows, select the ABAP data flow you want to add, and drag it into the workspace.
- To insert a new ABAP data flow, select the ABAP data flow icon in the tool palette, and click in the workspace.

Description

An ABAP data flow extracts data from SAP application sources. You can insert an ABAP data flow into any other data flow (ABAP, batch, or real-time). An ABAP data flow produces a data set you can use as a source in other data flows.

You can define parameters to pass values into the ABAP data flow. If you cannot express the required extraction logic using the Data Services interface, you can embed ABAP logic into an ABAP data flow by creating a new, custom ABAP transform.

When Data Services executes ABAP data flows, it translates extraction requirements into ABAP programs and passes them to SAP to execute. An ABAP data flow can store the extracted data set in an SAP application table or in text file saved on the SAP application server. To use the data outside the ABAP data flow, save the data in a file and use a data transport object to pass the data to the parent data flow. The data transport object defines a file format and indicates how data is transferred from the SAP application server to Data Services.

When you use the ABAP data flow as a source in a join (either with other ABAP data flows, SAP sources, or with non-SAP sources), you can assign a join rank to the source. When joining sources, Data Services reads sources with higher join ranks before reading those with lower join ranks. Set join rank in the ABAP data flow properties when you first define the ABAP data flow.

Note:
Join rank must be a positive integer. Zero is the default value. With a zero join rank, Data Services determines join order.

An ABAP data flow can contain the following objects:

- Sources
- Targets
- Transforms (Custom ABAP or Row_Generation)
- Queries
- ABAP data flows

ABAP data flows have the following options on the **Options** tab.

Option	Description
Datastore	Specifies the datastore corresponding to the SAP application tables or files this ABAP data flow accesses.
Generated ABAP file name	Specifies the file name containing ABAP program code that Data Services generates from this SAP ABAP data flow. This file is written to the directory specified in the datastore definition.
ABAP program name	Specifies the name Data Services uses for the ABAP program name that runs in SAP ABAP. It must begin with Y or Z.
Job name	Specifies the name used for the job that runs in SAP. It defaults to the name of the data flow.
ABAP row limit	Limits the number of rows that will be read by this ABAP data flow. Use this option for testing purposes to limit the time that the parent batch job will run. This option can also be used to limit the number of rows read using the Designer's Data Scan feature. ABAP data flows return one file of data from scans. Individual objects inside an ABAP data flow cannot be read using Data Scan.
Join rank	Indicates the rank of the table relative to other tables joined in a data flow. The software joins tables with higher join ranks before it joins tables with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
Parallel process threads	Specify the number of threads for parallel processing, which can improve performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option.

Option	Description
Cache	Indicates whether Data Services should cache the data set produced by this ABAP data flow for use in other operations. If the result is part of another ABAP data flow, the cache option is ignored. Best practice is to specify the cache only in the Query transform editor. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. The default option in the Query editor is Automatic and in a table source is No. The automatic setting in the Query editor carries forward the setting from the source table. **Note:** If any one input schema has a cache specified in the Query editor, the Data Services Optimizer considers only Query editor cache settings and ignores all source editor cache settings. **Yes**: The table is always cached unless it is the outer-most table in a join. **No**: The table is never cached. **Automatic**: Data Services determines whether to cache the table. The default option is **No**.

ABAP data flows have the following properties on the **General** tab:

Attribute	Description
Name	The name of the object. This name appears in the object library and in calls to the object.
Always execute during recovery	Indicates that Data Services should re-execute this ABAP data flow during an automatic recovery when it is part of a re-executed work flow that is a recovery unit. In this case, Data Services re-executes the ABAP data flow even if it completed successfully in an earlier run.
Description	Your description of the ABAP data flow.

Related Topics

- Creating custom ABAP transforms
- Adding an ABAP data flow

7.1.2.2 Custom ABAP transform

Class

Reusable

Access

- In the object library, click the **Transforms** tab then right-click and select **New ABAP Transform**. After renaming your custom ABAP Transform, drag it into an ABAP data flow. From the **Tools** menu, select **Variables** to define parameters.
- You can reuse an existing transform by dragging and dropping the object from the **Transforms** tab.

Description

A custom ABAP transform executes a piece of ABAP logic to produce a set of rows to be used by subsequent transforms in the same ABAP data flow.

A custom ABAP transform does not take any input sets and produces one and only one output data set. It allows for any number of input parameters, but no output parameters.

A custom ABAP transform has the following properties:

Properties	Description
Name	Specifies the name for the custom ABAP program that appears in the object library and in diagrams. This name cannot contain spaces.
Description	Allows you to document the transform's operation.

A custom ABAP transform has the following options:

Option	Description
ABAP language file name	Indicates the name of the file containing the ABAP program to execute.

Option	Description
Join rank	Indicates the rank of the output data set relative to other data sets included in a join. The software joins data sets with higher join ranks before it joins data sets with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see the *Data Services Reference Guide*.
Output schema	Lists the columns in the schema produced by this ABAP program. To add a column to the schema, right-click and choose **New Column**.
Column type	Indicates the data type of the selected column. This option appears after you create a column, click outside the column name, and then select the column.

7.1.2.3 Data transport

Class
Single-use

Access
- To insert, open an ABAP data flow diagram in the workspace, select the data transport icon in the tool palette, and click on the workspace.
- To view options, click the name of the data transport. This opens the data transport editor.

Description
Data transports are targets in ABAP data flows. A data transport takes the data SAP extracts during an ABAP data flow and stores it in a file defined by the Transport_Format file format.

Use a data transport when you want to use the extracted data in a data flow outside the ABAP data flow. In this case, you must use the ABAP data flow as a source in the parent data flow.

Do not use a data transport when you want the extracted data kept in a file on the SAP application server or in an SAP table. In that case, use a file or SAP table as the target in the ABAP data flow.

The data transport editor shows the schema produced by the ABAP data flow and the options that define the data transport. These options specify the location of the file and indicate whether to append to or overwrite the data in the file.

You can set the delimiter SAP uses to write data to the file. The default delimiter is a tab character. Change the delimiter in the Transport_Format file format definition. Valid ascii character delimiters range from /0 to /254.

The following table lists the options in the data transport editor:

Data transport editor option	Description
File Name	The name of the saved staging file.
Working directory on SAP	The location of the saved staging file. This is the **Working directory on SAP server** specified for the datastore associated with the current ABAP data flow.
Append file	Adds the data set to the existing contents of the file.
Replace file	Clears the file before writing the data set to it.

Data transports have one attribute:

Attribute	Description
Name	The name of the object. This name appears on the object in the diagram.

7.1.2.4 Extractor source

Class

Single-use

Access

To insert an SAP extractor source into a data flow, open the local object library, select the **Datastores** tab, select the extractor from the SAP Applications datastore, and drag it into the workspace.

Description

An SAP extractor is a source object that SAP BusinessObjects Data Services uses to read data from an SAP application.

Data Services automatically detects whether the extractor uses the ODP data replication API, which provides optimal performance and changed-data capture (delta) capability. Typically you use ODP data source extractors in a regular data flow, where Data Services uses RFC to call the ODP API. However if you want to do additional processing such as lookups and joins or for more advanced applications such as debugging or pushdown optimization, you can use them in an ABAP data flow, where Data Services generates the ABAP that calls ODP API.

Extractor option	Description
Name	Name of the extractor as defined in the datastore.
Type	Identifies the extraction type as defined by SAP. Examples: TRAN: Transaction ATTR: Attribute
Method	Identifies the extraction type as defined by SAP. Examples: V: Transparent table or database view F1: Function module (complete interface) F2: Function module (simple interface)
Datastore	The SAP Applications datastore name.
Datastore type	The type of SAP datastore (SAP Applications).
Application server	Name of the remote SAP application computer to which Data Services connects.
Language	Select the login language from the possible values in the drop-down list. You can enter a customized SAP language in this option. For example, you can type S for Spanish or I for Italian.
Client number	The three-digit SAP client number. Default is 800.

Extractor option	Description
System number	The two-digit SAP system number. Default is 00.
Join rank	Indicates the rank of the source relative to other tables and files in the data flow when creating a join. The software joins sources with higher join ranks before it joins sources with lower join ranks.
Package size	Limits the number of rows brought into memory when building internal tables. By limiting the number of rows, less memory is used to construct internal tables.
Initial load	For changed-data capture (CDC), indicates whether to reload all the data. Select to return a full extraction; clear to return only changed data. If you change the filter properties for an extractor, "reset" the job by running it with the source option **Initial load** selected. Then you can run the job with CDC filters applied (with **Initial load** cleared).
Extract from datetime	For changed-data capture (CDC), optionally enter a global variable of type datetime that defines from when to extract changed data. If the datetime value is the same as or before the value from the last execution, Data Services repeats the changed-data extraction. If the datetime value is later than the value from the last execution, Data Services returns the new data. Example: Yesterday the job ran with a datetime value of 2010.10.28 03:00:00. To reload the same data again (for example because of a problem with the execution), use the same datetime value again. To get new changes, use a later value, for example 2010.10.29 03:00:00.
Parallel process threads	For extractors that use the Operational Data Provider data replication API (ODP data replication API), specify the number of threads used to read the data. For example, if you have four CPUs on your Job Server computer, enter the number 4 for this option to maximize performance.

Related Topics

• Reading from SAP extractors

7.1.2.5 Hierarchy

Class

Reusable

Access

In the object library, click the **Datastores** tab and expand a datastore listing (click the plus sign next to the datastore name).

Description

Hierarchy objects allow you to browse, select, and import hierarchical information from a source database. Hierarchy objects can import the following hierarchical information:

Source application	Hierarchies extracted
SAP application	Data Services represents SAP application sets as hierarchies. In SAP applications, sets are group of objects such as cost element groups or cost center groups. The sets allow you to see an individual object or all similar objects in a related area. Hierarchy objects can import set information from SAP applications basic or single-dimension sets. In particular, you can import cost element and cost center hierarchies.

To extract hierarchical information, select a hierarchy type, import its metadata into your repository, then drag the hierarchy icon from the object library into a data flow.

SAP application hierarchies

External view

Hierarchy types are listed in the external browsing view for an SAP Applications datastore. From this view, you can choose to import a hierarchy type from a specific controlling area, or you can import a hierarchy type regardless of its association with a controlling area.

You can also import a hierarchy by specifying the hierarchy type by name. Select the datastore in the object library, right-click and choose **Import By Name**.

Imported hierarchy type

After you import the hierarchy type, it appears in the object library nested under the datastore name.

The hierarchy object has the following properties:

Property	Description
Name	The name of the hierarchy object. The name is constructed as follows: `set_table-hierarchy_group(datastore)`
Description	The description of the hierarchy type as included in the SAP application table.
Controlling_Area	The controlling area from which you imported the hierarchy type. You could specify the controlling area by selecting the hierarchy type in the controlling area view or by entering the controlling area name in the Import by Name window when you imported the hierarchy type. If you did not specify a controlling area, the value in Controlling_Area is an asterisk (*). If you import a chart of account hierarchy type, the controlling area will be inferred. The Controlling_Area value will never be NULL.
Set_Table	The table from which the hierarchy data is extracted.
Chart_Of_Account	The chart of account from which you imported this hierarchy type. This value is NULL for many hierarchy types associated with controlling areas but not chart of accounts.
Hierarchy_Group	The hierarchy type imported.
Check_Table	The location of the range of values for the hierarchy type leaf nodes.
Domain_Name	The data type and range definitions of the leaf node values.

Selecting the hierarchy in the object library and choosing **Open** displays the object properties and the source schema that results from the hierarchy extraction. Each row of the hierarchy object output describes a single relationship in the hierarchy.

Column name	Description
Parent_id	The parent node in the relationship described by this row.
Parent_desc	A text description of the parent node.
Child_id	The child node in the relationship described by this row.
Child_desc	A text description of the child node.

Column name	Description
Value_from	The start of the range of values associated with a leaf node. The range is found in the check table, which is recorded in the hierarchy object attributes. NULL if the row does not describes a leaf node.
Value_to	The end of the range of values associated with a leaf node. The range is found in the check table, which is recorded in the hierarchy object attributes. NULL if the row does not describes a leaf node.
Leaf_flag	A value of 1 indicates that the child node is a leaf node. A value of 0 indicates that the child node is not a leaf node.
Root_flag	A value of 1 indicates that the parent node is the root node. A value of 0 indicates that the parent node is not the root node.
Tree_level	The level of the child node in the hierarchy relative to the root node.
Cont_area	The controlling area associated with the hierarchy type. If this hierarchy type was selected without specifying a controlling area, the value of Cont_area is an asterisk (*).
Chart_of_acct	The chart of account associated with the hierarchy type.
Set_table	The table from which the hierarchy data is extracted.

Note:
For every root node in the hierarchy, the hierarchy object produces a row with NULL as the root node. Depending on the final use of the hierarchy data, you may want to filter the NULL parent rows out of the hierarchy data.

Hierarchy instance

You can drag a hierarchy from the object library into an ABAP data flow definition. The hierarchy object appears in the data flow definition as follows:

The hierarchy editor displays the datastore information for the hierarchy and provides options for the instance of the hierarchy object:

Option	Description
All root nodes or Root node	Indicates that all hierarchies (root nodes) will be extracted from the specified hierarchy type. Turn off this option to specify a single hierarchy to extract. Look in the external view of the datastore to determine specific hierarchy names.
All controlling areas or Controlling area	Indicates that all hierarchy types for all controlling areas in which the groups appear will be extracted. Turn off this option to specify a single hierarchy type for a single controlling area to extract. **Note:** For best performance, make sure to specify a single controlling area.

Related Topics

- Designer Guide: Datastores, Importing metadata through a database datastore

7.1.2.6 IDoc file

Class

Reusable

Access

In the object library, click the **Datastores** tab, click an SAP Applications datastore, and click the list of available IDocs.

Description

IDoc file sources are handled the same way in data flows of real-time and batch jobs.

See Source for information about IDoc source message and file attributes.

See Target for information about IDoc target message and file attributes.

A real-time job that contains an IDoc file source processes its files one at a time. A real-time job that contains an IDoc message source can process messages one at a time or in parallel. See IDoc message for more information.

Multiple File Read in batch jobs

One or more IDoc files can be saved into one file or into many individual files to be processed at a later time.

When you run a real-time job in Test mode, Data Services can read multiple IDoc files at one time. To configure Data Services to read files in groups, list the names of these IDoc files or use one name with wildcard characters in the **IDoc Files** box of the source editor in the Designer.

The following list provides three examples:

- If you specify the IDoc file D:\temp\IDOC_SAP*.txt, Data Services will process data from all files in the D:\temp directory with names beginning with IDOC_SAP and the extension .txt.
- If you specify the IDoc file D:\temp\IDOC_SAP?.txt, Data Services will process data from all files in the D:\temp directory with:
  - names beginning with IDOC_SAP
  - any character before the extension
  - the extension .txt.
- If you specify the IDoc file D:\temp\IDOC_01.txt, D:\temp\IDOC_02.txt, D:\temp\IDOC_03.txt, Data Services will process data from these three IDoc files.

You can identify the source file for each row by using the Include file name column. For details, see Source.

IDoc file sources are validated against the IDoc Name, its related hierarchy, and the SAP release.

Variable File names in batch jobs

You can also use variable file names in the **IDoc File/s** box. Define a global or local variable for an initialization script of a job. For example,

```
$SOPGEN01_filepath = "D:\temp\IDOC_SAP*.txt"
```

Then enter *$filepath* in the **IDoc Files** box. This way you can avoid drilling down to each IDoc object to configure the location of source files.

7.1.2.7 IDoc message

Class

Reusable

Access

In the object library, go to the **Datastores** tab, open the SAP Applications datastore, and open the list of available IDocs.

Description

An IDoc is an intermediate document that an SAP application produces to communicate transaction data between SAP application servers. SAP applications offers numerous IDocs that describe different transaction types. The data in an IDoc supports a particular transaction. An IDoc can contain data from multiple tables and can contain hierarchical data.

You can use an IDoc message as a source or target in a real-time job. You can use an IDoc message as a target in a batch job. You cannot use an IDoc in an ABAP data flow.

You can use the multiple file read and variable file name features with IDoc message sources in real-time jobs. However, these features can only be used when the job is run in test mode.

IDoc message sources must be configured in the Data Services Administrator as well as in Data Services Designer.

Real-time jobs that contain an IDoc message source can be processed one at a time or in parallel. To enable parallel processing, select **Administrator** > **Real-time** > **Client Interface** in the Data Services Administrator.

The **Parallel Processing** option allows you to increase the number of IDoc source messages processed per minute for the IDoc type specified. This option is disabled by default.

The parallel processing option allows the Access Server to send an IDoc to a service queue (where it waits for a service provider) and continue with the next IDoc without waiting for reply. The maximum number of outstanding IDoc requests in the queue is the number of IDocs received or 4, whichever is smaller.

Note:
Where a strict IDoc processing sequence is required, do not use the Parallel Processing option.

To use IDocs, you must first import the metadata from the SAP application to your repository. Data Services imports the schema for the IDoc, maintaining hierarchical relationships among the data fields. You can display the IDoc schema by double-clicking the IDoc name in the object library.

When IDoc metadata is imported into Data Services or when IDocs are sent and received, the control data appears as the first segment for all IDoc objects. The control data is commonly prefixed with EDI_DC. The control data contains IDoc administrative information, such as IDoc number and sender or receiver information.

Related Topics

- Importing IDoc metadata using the Search option
- Source
- Target
- IDoc file
- Creating real-time jobs using IDoc sources

7.1.2.8 Open Hub Table source

Class

Single-use

Access

To insert an Open Hub source into an SAP BusinessObjects Data Services data flow, open the local object library, go to the **Datastores** tab, select the Open Hub table from the SAP BW Source datastore, and drag it into the workspace.

Description

An Open Hub Table is a source object that SAP BusinessObjects Data Services uses to read data from an SAP NetWeaver Business Warehouse system.

Note:
You can only use an Open Hub Table as a source object, and it can only be used in a regular data flow. You cannot use it in an ABAP data flow.

The following options are available for an Open Hub source.

Open Hub source option	Description
Make Port	Makes the source table an embedded data flow port. For more information, see “Embedded Data Flows” in the *SAP BusinessObjects Data Services Designer Guide*.
Join rank	Indicates the rank of the source relative to other tables and files in the data flow when creating a join. The software joins sources with higher join ranks before it joins sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in the source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see “Other Tuning Techniques” in the *Data Services Performance Optimization Guide* .

Open Hub source option	Description
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. There are two options: • **Yes**: The source is always cached unless it is the outer-most source in a join. • **No**: The source is never cached. The default is **No**. A cache setting specified in the Query transform editor FROM tab overrides any cache setting specified in a source. Best practice is to specify the cache setting only in the Query transform editor. **Note:** For data flows that you created prior to version 11.7, the default cache type is in-memory. If the data retrieved does not fit into available memory, change the cache type to pageable in the data flow "Properties" window.
Open Hub Table	Specifies the name of the Open Hub table to read from. The default value is the name of the current Open Hub table. You can do any of the following: • Leave the default value if different people are not using it simultaneously. • Select another Open Hub table name from the drop-down list, but the one you select must have the same schema as the current Open Hub table. • Select a substitution parameter from the drop-down list, but the one you select must have the same schema as the current Open Hub table. You must define the substitution parameter before creating this data flow. This capability allows you to reuse the data flow to read from another Open Hub destination that obtains its data from the same InfoProviders with the same schema. For more details, see the Related Topics below.
Execute process chain before reading	Specifies whether or not to execute the process chain in the BW system before this Data Services data flow reads data from the Open Hub table. The process chain reads data from the BW InfoProvider objects and loads the Open Hub table from which the Data Services Open Hub source object reads the data. By default, this option is not selected. In this case, you must start the process chain with the function sap_openhub_processchain_execute in a script.

Open Hub source option	Description
Process chain name	Specifies the name of the process chain in the BW system for this Open Hub table when **Execute process chain before reading** is selected. If you select **Execute process chain before reading**, the software fills in this option with the first process chain in the attributes for the current Open Hub table. Ensure that this is the appropriate process chain. If you change the name of the Open Hub table, you can select the corresponding process chain name from the drop-down list. You can specify a substitution parameter that you define before creating this data flow.
Parallel SAP Connections	Specifies the number of Open Hub connections to read multiple data packets from a BW system simultaneously. The default value is 0. Valid values are 0 and any positive integer. A value of 0 or 1 disables parallel reading. You can specify a substitution parameter that you define before creating this data flow.

Related Topics
- Creating a data flow for multiple users to read one InfoProvider

7.2 Functions

SAP BusinessObjects Data Services does not support functions that include tables as input or output parameters except RFC and BAPI functions imported from SAP Applications.

7.2.1 SAP application RFC and BAPI function calls

You can call SAP application Remote Function Call (RFC) enabled functions, including Business Application Programming Interface (BAPI) functions, from queries inside data flows.

To make an RFC function available to call from Data Services data flows, import the metadata for the function from the SAP application server using an SAP Applications datastore connection. Be aware that the requirements for RFCs and BAPIs, and therefore their metadata, may be different between versions of SAP applications.

If you design data flows with BAPI calls against one version of an SAP application, then change datastores to a later version of SAP, Data Services allows this without the need to reimport the BAPI. Any new parameters added to the function call, including additional columns in table parameters, are added automatically to the call and filled with NULL values. Thus Data Services allows you to design jobs that are portable between SAP systems.

For a Data Services job to execute an RFC function, the login indicated by the datastore into which you imported the function must include the appropriate permissions required to execute the functions.

After you import the metadata for an SAP function, the function is listed in the Functions category of the SAP Applications datastore. You will also see the function in the function wizard listed under the datastore name.

When a function can provide a single output value, you can call the function in a column mapping in a query. Data Services provides a function wizard (available by clicking the **Functions** button on the **Where** tab of a query) to help you include input parameters for the function call. Input parameters can include expressions.

When a function can provide more than one output value, you can call the function by including it as an object in the output schema of a query. Data Services provides a function wizard (available through a short-cut menu) to help you include input and output parameters for the function call.

Data Services supports tables as input and output parameters for SAP RFC and BAPI functions. The function import process automatically includes the metadata for tables included as function parameters.

To specify a table as an input parameter to a function, the table must be an input to a query, either as a top-level input or nested under the top-level. The table must also be available in the FROM clause of the context where you call the function. Data Services maps columns in the input schema by name to the columns in the table used as the function input parameter. You need only supply the columns that are required by the function. At validation, if Data Services encounters type mismatches between supplied columns and the function signature, it attempts to convert the given type to the expected type. For type mismatches that it cannot resolve, Data Services produces validation errors.

One of the values that an RFC function can return is AL_RFC_RETCODE. This column contains a flag that identifies the success or failure of the function call. The possible values for AL_RFC_RETCODE are as follows:

Value	Description	Returned by
BOBJ_DI_RFC_OK	The RFC call succeeded. This value is replaced by the return value from the RFC call.	Data Services
BOBJ_DI_RFC_ENABLE_ERROR	The function is not RFC enabled.	Data Services
BOBJ_DI_RFC_NOT_ENABLED	The function is not RFC enabled.	Data Services

Value	Description	Returned by
BOBJ_DI_RFC_CALLRE CEIVE_ERROR	The function call in SAP returned an error.	Data Services
BOBJ_DI_RFC_R3_CONN_EXCEP TION	The SAP Applications datastore connection cannot be created, because of a connection error, invalid user, password, system number, or host name.	Data Services
BOBJ_DI_RFC_CALL_ERROR	The connection completes, but the call fails in SAP.	Data Services
BOBJ_DI_RFC_GET_RESULT_ER ROR	Data Services cannot obtain the result of the function call from SAP.	Data Services
BOBJ_DI_RFC_COMMIT_ERROR	Data Services cannot commit the work because the BAPI_TRANSACTION_COMMIT call returned an error.	Data Services
BOBJ_DI_RFC_BAPI_NOT_SAME	The Data Services repository contains a different BAPI function than in SAP.	Data Services
RFC_OK	The function call succeeded. Look for the results or errors that it returns.	SAP application
RFC_FAILURE	The function call returned an error. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP application
RFC_EXCEPTION	The function call returned an error. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP application

Value	Description	Returned by
RFC_SYS_EXCEPTION	The function call returned an error and closed the connection to Data Services. If the function is a BAPI, details for the cause of the error are available in the RETURN structure available as an output from the function.	SAP application
RFC_CALL	The function call was received by SAP. If this value is left, the function failed to return a success flag after starting.	SAP application
RFC_INTERNAL_COM	An internal communication error occurred within SAP.	SAP application
RFC_CLOSED	The SAP application closed the connection and cancelled the function call.	SAP application
RFC_EXECUTED	The SAP application already executed the function call.	SAP application
RFC_MEMORY_INSUFFICIENT	The SAP application does not have enough memory available to process the function call.	SAP application
RFC_NO_TID	The transaction ID is not available to SAP.	SAP application
RFC_RETRY	The SAP application did not process data yet. SAP will retry the function call.	SAP application
RFC_NOT_FOUND	The SAP application could not find the function.	SAP application
RFC_CALL_NOT_SUPPORTED	The SAP application does not support the function call.	SAP application

Value	Description	Returned by
RFC_NOT_OWNER	The login in the Data Services datastore cannot connect to SAP.	SAP application
RFC_NOT_INITIALIZED	The Data Services RFC library did not initialize properly.	SAP application
RFC_SYNCHRONIZE	The SAP application is busy processing a synchronous call.	SAP application
RFC_SYSTEM_CALLED	Data Services is busy executing a call from SAP.	SAP application
RFC_VERSION_MISMATCH	The version of the function call from Data Services is incompatible with the function expected by SAP.	SAP application

The RETURN structure for BAPIs varies between releases of SAP applications:

Field	Description
TYPE	blank — success S — success E — error W — warning I — information A — abort
CODE	Error message numbers
MESSAGE	Error message text in the language chosen at login

This TYPE value is blank or NULL depending on the current setting of the Server option **Convert SAP null to null**. Check this option by choosing **Tools** > **Options** in the Designer. In particular when calling BAPI functions, the data you provide through the BAPI call might be different from the data that you use to test a BAPI directly in the SAP GUI interface. The SAP application interface automates data handling, where the BAPI operation undercuts the interface level.

Consider the following issues:

- All character values must be uppercase
- Padding values
- Assumed decimal values (QTY)
- Codes are language-specific
- Automatic type conversion
- SAP application version-specific behavior

To determine the data requirements of various SAP application functions, you can read the function requirements in the SAP GUI transaction screens:

- BAPI list by functional area: `bapi`
- BAPI and RFC source and input and output parameters: `se37`

You can also determine appropriate values, such as the language-specific code values, by looking at the table where the data is ultimately stored.

Related Topics

- Designer Guide: Real-time Jobs, Calling application functions
- Reference Guide: Data Types, Type conversion

7.2.2 Data Services built-in functions

This section describes the Data Services functions that are specific to SAP.

Function	Category	Description
lookup	Lookup	Finds a value in one table or file based on values in a second table or file.
sap_extractor_delta_initialize	SAP	Sets a marker from which to return changed data.
sap_openhub_processchain_execute	SAP	• Starts the process chain that extracts data from an InfoProvider (InfoArea, InfoCube, or DataStore object) on SAP NetWeaver Business Warehouse and loads the extracted data into an Open Hub Destination table. • Monitors the process chain status and the Open Hub Destination request notification.

Function	Category	Description
sap_openhub_set_read_status	SAP	Sends the read status for the Open Hub table to SAP NetWeaver BW. A read status causes SAP NetWeaver BW to release the Open Hub Destination table so that other users can access it.
substr	String	Returns a specific portion of a string starting at a given point in the string.
sy	Miscellaneous	Returns the value of an SAP system variable at run time. This function is only available through query transforms in ABAP data flows.

7.2.2.1 lookup

For SAP application tables, no owner value is required, but the periods are. For example:
`sap_ds..KNA1`

`cache_spec` : The caching method the lookup operation uses. List within single quotes.

With regard to the three possible settings:

- NO_CACHE

  Reads values from the lookup_table for every row without caching values.

- PRE_LOAD_CACHE

  For SAP application tables, applies the filters and loads the result_column and the compare_column into an internal SAP table. The function reads the values from the internal table.

  Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values.

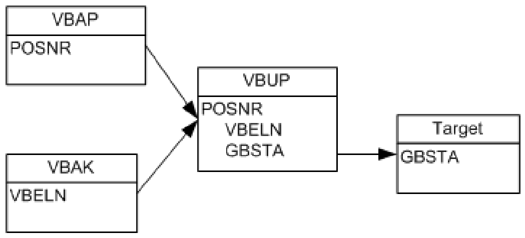
- DEMAND_LOAD_CACHE

  Select this option if the number of rows in the table is large and you expect to access a low percentage of the table values frequently.

  For SAP application tables, loads the result_column and compare_column into an internal SAP table when two or more lookups use the same translation table. So, Data Services creates a single SELECT statement to load the combined values from all the lookups into an internal SAP table.

  Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

You can use the lookup function when retrieving data from SAP application tables in an ABAP data flow. For example, suppose you are joining two tables, VBAK and VBAP, and you want to retrieve a value from a third table, VBUP.



You can create a lookup function with two `expression` and `compare_column` pairs:

```
lookup(sap_ds..VBUP, GBSTA, 'none', 'NO_CACHE', VBELN, VBAK.VBELN, POSNR, VBAP.POSNR)
```

This function returns the value from the GBSTA column (the `result_column`) in the VBUP table (the `lookup_table`) that corresponds to the VBELN value (first `compare_column`) in the VBAK table (first `expression`) and the POSNR value in (second `compare_column`) the VBAP table (second `expression`). When no corresponding value is found, the function returns "none."

When there are multiple matching rows in the `lookup_table` and the lookup table is an SAP application table, the lookup function randomly selects a matching row and returns the value in the result_column for that row. To avoid random selection:

- Use the lookup_seq function if the table has a sequence column that discriminates between matching records
- Use an outer join and add criteria to the WHERE clause to discriminate between multiple matching records

7.2.2.2 sap_extractor_delta_initialize

For SAP extractors using changed-data capture, use this function in a script to indicate to the ODP data replication API to only extract changed data (delta) for subsequent executions.

```
sap_extractor_delta_initialize (datastore_name, extractor_name, $g1)
```

where $g1 is a sample name for a global variable of type string. After calling the function, $g1 is empty to indicate success; otherwise, it displays E for error.

7.2.2.3 sap_openhub_processchain_execute

Performs the following tasks:

- Starts the process chain that extracts data from an InfoProvider (InfoArea, InfoCube, or DataStore object) on SAP NetWeaver Business Warehouse and loads the extracted data into an Open Hub Destination table.
- Monitors the process chain status and the Open Hub Destination request notification.

When the function returns successfully, an Open Hub table source in SAP BusinessObjects Data Services can then read the data from the Open Hub Destination table.

Note:
You can only use this function in a script. It is not valid in a query or audit object.

Below is the function syntax as a reference. The function wizard is explained in the next section.

Syntax

```
sap_openhub_processchain_execute('datastore','open_hub_table',
'process_chain',$logid,$ReturnTxt)
```

Where

datastore	Specifies the datastore name. You can specify either a constant string or a substitution parameter. The data type is varchar(64).
open_hub_table	Specifies the Open Hub Destination table. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
process_chain	Specifies the name of the process chain that extracts data from the InfoProvider in SAP NetWeaver BW and loads the data to the Open Hub Destination table. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
$logid	(Optional) Specifies a variable to obtain a value that depends on the function return value (see "Relationship between return value and value of logid variable" below). The required variable data type is varchar(25).
$ReturnTxt	(Optional) Specifies a variable to retrieve the description of the return status of the process chain. The required variable data type is varchar, and you can define the length you want for this variable.

Return value

varchar(1)

Returns one of the following values.

Return value	Description
B	Open Hub Destination is being read by another user.
E	Data Services error while executing the function.
R	Process chain execution failed with errors in BW system.
X	Process chain execution has been canceled in BW system.
S	Function successfully executed the Open Hub extraction Data Transfer Process (DTP) and received extraction request notification.

Relationship between return value and value of logid variable

The value of the logid output variable depends on the function return value, as the following table shows.

Return value	$logid variable value	$ReturnText variable value	Action
B	Process chain log ID of the other user that is currently reading the Open Hub Destination	Status of current Open Hub extraction	Either wait and try again or stop executing the data flow that contains the Open Hub Destination table.
E	Data Services error log number	Data Services error text	Stop executing the data flow that contains the Open Hub Destination table, and analyze the Data Services error.
R	Your process chain log ID	Error from process chain	Stop executing the data flow that contains the Open Hub Destination table, and use the log ID in the BW system to see the detail state of the process chain error.
X	Your process chain log ID	Error from process chain	Stop executing the data flow that contains the Open Hub Destination table, and use the log ID in the BW system to see the detail state of the process chain error.

Return value	$logid variable value	$ReturnText variable value	Action
S	Open Hub extraction request ID	Status of your Open Hub extraction	Use the request ID in the BW system to obtain detail loading statistics (such as number of packets loaded and number of records loaded).

Example:

The following sample script commands check the return value, generate an exception, and print the error if the function is not successful.

```
$status = sap_openhub_processchain_execute('open_hub_datastore',
'Materials','Materials_PC',$lpcogid,
$returntxt);
If ($status != 'S') raise_exception ('Materials_PC process chain execution failed ' || $returntxt);
```

Restrictions

The following are restrictions for using Open Hub Destinations:

- Only one job at a time can read an Open Hub Destination table.
- A process chain of an Open Hub Destination can contain only one of its Data Transfer Processes (DTPs).
- A process chain cannot contain DTPs for more than one Open Hub Destination.

7.2.2.3.1 To define an sap_openhub_processchain_execute function

1. To access the function wizard for sap_openhub_processchain_execute from the Script Editor, click **Functions** or **...** at the top of the window.
2. Select **sap_openhub_processchain_execute** from the list of functions.
   The "Define Parameters" window opens.
3. Select an SAP BW Source datastore name from the drop-down list. You can also select a substitution variable from the list. If you type in a datastore name, put single quotes around the name.
4. Select the name of a Open Hub table from the drop-down list. Only the names of the imported Open Hub Tables appear in this list.
5. Select the name of a Process Chain from the drop-down list.
6. Specify a variable that will get the BW log ID for the process chain after the function executes. You must define the variable before you can use it.
7. Specify a variable that will get the description of the status after the function executes. You must define the variable before you can use it.

7.2.2.4 sap_openhub_set_read_status

Sends the read status for the Open Hub table to SAP NetWeaver BW. A successful read status causes SAP NetWeaver BW to delete the data from the Open Hub Destination table.

Syntax

```
sap_openhub_set_read_status('datastore','destination',
status,$returntxt)
```

Where

datastore	Specifies the datastore name. You can specify either a constant string or a substitution parameter. The data type is varchar(64).
destination	Specifies the Open Hub Destination in the BW system. You can specify either a constant string or a substitution parameter. The data type is varchar(30).
status	Specifies the read status. Possible values are either a variable or one of the following string constants: • 'X' for Read Successful • Any other value indicates that the Read failed. The data type is varchar(1).
$returntxt	(Optional) Specifies a variable to return the status log of the function. The required variable data type is varchar, and you can define the length you want for this variable.

Return value

varchar(1)

Returns one of the following values.

Return value	Description
S	Success
E	Error

Example:

The following sample script commands sends the status of the Open Hub table read to the BW system and prints the status.

```
$status = sap_openhub_set_read_status('BR9", 'PS_BOOK_5', 'X', $returntxt):
print ('Status: ' || $status);
```

7.2.2.5 substr

In an ABAP data flow, the first character is position 0; in other data flows, the first character is position number 1. If *start* is 0 in a non-ABAP data flow, the new string begins with the first character (that is, position 1). If *start* is negative, the new strings begins with the first character in the string (that is, position 0 in an ABAP data flow or position 1 in a non-ABAP data flow).

7.2.2.6 sy

Returns the value of an SAP system variable at run time. This function is only available through query transforms in ABAP data flows.

Syntax

```
sy('SAP_variable')
```

Return value

varchar(255): The value of the SAP system variable. You may need to recast the return value to the actual data type of the system variable in SAP.

Where

SAP_variable: A string value containing the name of the SAP system variable. This value is not case sensitive. Enclose the name in single quotation marks (').

When the sy function is executed, Data Services generates the appropriate function call in the ABAP for the ABAP data flow (appends `SY-` to the *SAP_variable* that you specify) and returns the result of the function call at run time.

The table SYST in SAP lists the available system variable, their data types, and descriptions.

If the given *`SAP_variable`* does not exist in SAP, Data Services produces a run-time error:

```
ABAP program <Generated ABAP Program> syntax error: <The data object "SY" has no component called "ABC">.
```

No value is returned when this error occurs.

Example: Functions and results

Function	Results
`sy('SUBRC')`	0 if the subroutine call returned successfully
`sy('mandt')`	The client ID number of the current SAP application client
`sy('UNAME')`	The logon name of the user
`sy('DATUM')`	The current date from the SAP application server
`sy('UZEIT')`	The current time from the SAP application server
`sy('TCODE')`	The current transaction code

7.3 Data types

The following sections discuss SAP application data types used in Data Services.

7.3.1 Conversion to/from Data Services internal data types

The following table lists the conversions from SAP application data types to Data Services internal data types.

Source	External data type	Converts to Data Services data type
SAP application	date (D)	date
	time (T)	time
	float (F)	double
	integer (I)	int
	numeric (N)	varchar
	packed (P)	decimal
	char (C)	varchar
	hexadecimal (X)	varchar

The following table lists the conversions from Data Services internal data types to SAP application data types.

Internal data type	Is converted to SAP application data type
date	date (D)
datetime	date (D)
decimal	decimal (P)
double	float (F)
int	int (I)
interval	int (I)
numeric	char (C)
real	float (F)
time	time (T)
varchar	char (C)

7.3.2 Working with null values from SAP applications

SAP applications use a different standard for null values than Data Services or the target databases that Data Services supports. You can configure Data Services to account for the difference in one of two ways:

- Always converting the SAP equivalent of a null value to the Data Services NULL constant
- Reading the SAP null value literally

Each method has consequences for how you can use null-value tests in analytic processing. Data in a given warehouse must be loaded over time using the same method.

The default configuration for Data Services is to read the SAP value literally. You can change this configuration in the Designer option **Convert SAP null to null**. Go to the **Tools** menu and choose **Options**. Under **Administrator Options**, choose **Server Options** to see the **Convert SAP null to null** check box.

7.3.3 Design considerations for SAP application null behavior

When choosing which method to use to handle SAP application null values and when designing extraction logic in Data Services, consider the following issues:

- Primary keys built from concatenated values

  Spaces are often used in SAP applications where null values would cause an error in a primary key built from several key values concatenated together. If you choose to have Data Services automatically convert these spaces to null values, you may have to include null-value replacement logic in your data flows to avoid errors in primary keys for extracted tables.
- Null date values

  If dates are left to Data Services and the target database conventions to determine the conversion of an SAP null date value, the result may not be useful for constructing reports. See Null dates for more details.
- Filters for null values

The value used to match an SAP null value depends on the data type of the value.

Data type	SAP application null value
Strings	single space
Numbers	zero
Dates	`00000000`

Data type	SAP application null value
Times	`000000`

7.3.4 Null dates

The concept of a null date in SAP applications is represented by a value called INITIAL that is a property of every SAP application data element. You can see INITIAL dates displayed through SAPGUI as `00/00/0000` (month 0, day 0, year 0).

In the default configuration of Data Services (where **Convert SAP null to null** is disabled), INITIAL dates are written to an ABAP data file as NULLs. In ABAP data files, NULLs are represented by empty strings. They are then read by the parent data flow, which converts every empty string to a single space. If passed through again without data type or value change, Data Services converts the single blank into 01-JAN-1900 before loading it into a target.

The single space is not a valid date value. When Data Services encounters a single space in date conversion operations (implicit or explicit) or when evaluating an expression, the results are not predictable. To avoid unpredictable results, the data flow must explicitly convert the INITIAL date to a valid date value.

Consider these techniques to convert INITIAL dates to valid date values:

- Convert INITIAL dates to valid dates using the nvl function in the ABAP data flow

  In an ABAP data flow, Data Services produces ABAP for the nvl function that looks for an INITIAL date rather than a null value. The mapping expression can be the following:

  ```
  nvl(date_column, valid_date)
  ```

  where the valid date is a parameter passed to the data flow or a string indicating the appropriate initial date for your data warehouse.

  An example mapping expression would be:

  ```
  nvl(VBAK.ANGDT, '19901231')
  ```

- Turn on the option **Convert SAP null to null** and use nvl in regular (non-ABAP) data flows

  If an INITIAL date is extracted from SAP and loaded directly into a Data Services transport file, the INITIAL date is converted to a null value. If the Convert SAP null to null option is turned on, the null values in the transport file are brought into the data flow as null values. You can then use the nvl function in the data flow to detect the null date and substitute an appropriate valid date for it.

Index