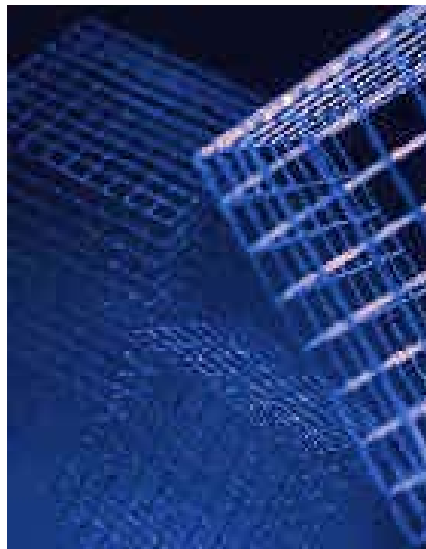


# Replication Manager: SAP DB



**Version 7.3**








## Copyright

© Copyright 2002 SAP AG.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation.

For more information on the GNU Free Documentaton License see  
<http://www.gnu.org/copyleft/fdl.html#SEC4>.

## Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

## Typographic Conventions

Type Style	Description
<i>Example text</i>	Words or characters that appear on the screen. These include field names, screen titles, pushbuttons as well as menu names, paths and options.  Cross-references to other documentation
<b>Example text</b>	Emphasized words or phrases in body text, titles of graphics and tables
EXAMPLE TEXT	Names of elements in the system. These include report names, program names, transaction codes, table names, and individual key words of a programming language, when surrounded by body text, for example, SELECT and INCLUDE.
Example text	Screen output. This includes file and directory names and their paths, messages, names of variables and parameters, source code as well as names of installation, upgrade and database tools.
<b>Example text</b>	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries.
EXAMPLE TEXT	Keys on the keyboard, for example, function keys (such as F2) or the ENTER key

Replication Manager: SAP DB .....	9
Using the Replication Manager.....	9
Loading Data to Multiple Tables Simultaneously .....	10
Loading Binary Values .....	11
Selecting Data Records from the Source File .....	12
Terminology .....	12
REPM Server .....	13
Command File .....	13
Structure of a Command File.....	13
General Syntax Rules for Command Files .....	13
Source File .....	14
Target File .....	15
Source Table .....	15
Target Table .....	15
Qualification Column .....	15
Remote Server .....	15
Data Types .....	15
External Data Types .....	16
Data Types Used Internally by the Database .....	16
Plain Text Values.....	17
Binary Values.....	17
HEX Values .....	17
Data File Formats.....	18
COMPRESSED .....	18
FORMATTED .....	19
FORMATTED BINARY .....	20
Log File of the Replication Manager .....	21
Calling the Replication Manager .....	21
Options.....	22
Name of Command File: -b .....	22
Name of the Database Instance: -d .....	23
Maximum Permitted Number of Errors: -E .....	23
Help for the Options: -h .....	23
Name of the Host Node of the Database Instance: -n .....	24
Output File of the Replication Manager: -o[w a].....	24
Substitution Parameter: -p.....	24
InstRoot Directory: -R.....	25
Name of the REPM Server Host Node: -r .....	25
User Data: -u .....	26

REPM Server Version: -V .....	26
Version of the Replication Manager: -v .....	26
REPM Server Commands .....	26
Commands for Setting Up a Session .....	27
AUTOCOMMIT Command .....	27
IGNORE TERMCHARSET .....	28
SET Command .....	28
SQLMODE Command .....	29
USE SERVERDB Command .....	29
USE TERMCHARSET Command .....	29
USE USER Command .....	30
USE USERKEY Command .....	30
VERSION Command .....	30
Commands for Loading and Unloading Data and Catalog Contents .....	31
FASTLOAD Command .....	32
DATALOAD Command .....	34
Examples of DATALOAD Commands .....	35
DATAEXTRACT Command .....	36
DATAUPDATE Command .....	37
TABLELOAD Command .....	38
TABLEEXTRACT Command .....	40
Syntax Rules for Commands .....	41
Syntax Rules for Setting up a Session .....	41
database_name_statement .....	41
sql_mode .....	42
user_statement .....	42
Syntax Rules for Table Descriptions .....	42
compare_operator .....	43
condition .....	43
duplicates_clause .....	44
field_format .....	44
field_pos .....	45
if_condition .....	46
restore_spec .....	47
simple_condition .....	47
table_name .....	48
table_spec .....	49
usage_spec .....	49
Syntax Rules for Column Descriptions .....	50
acc_column_spec .....	50
acc_column_spec_mlt .....	51

column_assignment .....	51
column_descriptor .....	52
column_id .....	53
column_id_spec .....	54
column_names .....	55
format_spec .....	55
generate_spec .....	56
key_column_spec .....	56
lit_column_spec .....	57
load_column_spec .....	57
load_column_spec_mlt .....	58
null_assign .....	59
null_condition .....	60
numerical_functions .....	61
order_clause .....	62
output_column .....	62
output_column_list .....	63
round_or_trunc_spec .....	64
scale_spec .....	64
select_expression .....	65
set_column_spec .....	66
set_column_spec_mlt .....	66
sequence_number .....	67
simple_column_spec .....	67
Syntax Rules for Describing Data Access .....	68
bool_spec .....	68
code_spec .....	68
date_spec .....	68
delimiter_spec .....	69
extract_files_spec .....	69
file_extract_spec .....	70
file_format_spec .....	71
infile_spec .....	71
int_spec .....	71
longfile_code_spec .....	72
longfile_spec .....	72
Loading LONG Values .....	73
Each LONG Value to Be Inserted in a Separate LONG Data File .....	73
All LONG Values to Be Inserted in One Data File .....	75
Unloading LONG Values .....	76
Each LONG Value to Be Unloaded in a Separate LONG Data File .....	77

All LONG Values to Be Unloaded Are in One File.....	78
longfile_spec_mlt .....	78
noheader_spec .....	79
number_spec .....	79
null_spec .....	80
outfile_spec .....	80
part_spec.....	80
separator_spec .....	81
standard_code_spec.....	81
standard_date_mask.....	81
standard_time_mask.....	82
standard_timestamp_mask.....	82
time_spec.....	83
timestamp_spec .....	83
SQL Statements .....	84
Keywords and Alternative Keywords.....	84
Keywords A - C .....	84
Keywords D - E.....	85
Keywords F - K .....	86
Keywords L - P .....	86
Keywords R - S.....	87
Keywords T - Z .....	87
Processing Commands .....	88
Processing Commands with the Replication Manager CLI .....	88
Using Perl .....	89
Perl Classes .....	90
Perl: RepMan Class.....	90
Perl: Exception Classes .....	91
Perl: Example No. 1 .....	91
Perl: Example No. 2 .....	92
Perl: Example No. 3 .....	93
Perl: Example No. 4 .....	94
Perl: Example No. 5 .....	95
Using Python .....	96
Python Classes .....	98
Python: RepMan Class.....	98
Python: Exception Classes.....	99
Python: Example No. 1 .....	99
Python: Example No. 2 .....	100
Python: Example No. 3 .....	101
Python: Example No. 4 .....	102

Python: Example No. 5 .....	104
-----------------------------	-----



## Replication Manager: SAP DB

You can use the Replication Manager for SAP DB databases with Version 7.2 and above.

The Replication Manager is a tool designed to allow developers and database administrators to load data from external files into SAP DB databases and to unload data from SAP DB databases into external files.

The Replication Manager is purely a batch tool. It can process individual statements or a sequence of statements, but is also offered as an extension for script languages Perl and Python. The batch file is a command file that contains the statements and commands that are to be executed. This command file also specifies the external data files.

The Replication Manager can process a great many input and output formats and allows users to load data into the database, or unload data out of a database. The data to be loaded can also be filtered, meaning it can be selected on the basis certain values.

As well as being able to process special load and unload commands, the Replication Manager can execute all SQL statements. This means that you can create command files that contain SQL statements and DATALOAD commands. The DDL functions of the SQL statements are particularly important for the load commands.

When it processes the commands, the Replication Manager generates a log file in which processing information is stored. If load or unload commands reject data records because of errors, these records are also entered in the log file.



### Using the Replication Manager

Using batch files, you can load data from external files into SAP DB database systems, or unload SAP DB database system data into external files.

There are two kinds of batch execution:

- Direct execution of a file with a sequence of commands and statements;  
if you want to execute a sequence of Replication Manager commands and SQL statements and no return values <> 0 occur
- Calling of statements to be executed from script languages (Perl or Python);  
if you want to react to the return values of a command or statement

You use the Replication Manager for the following activities:

Aim	Formats supported	Replication Manager commands
Load data from a data file to a SAP DB instance	CSV (comma separated values = COMPRESSED)  FWV (fixed width, columnar values = FORMATTED)  Compact (proprietary format)  SAP DB Page (proprietary format)	DATALOAD, FASTLOAD, TABLELOAD, DATAUPDATE
Unload data from a SAP DB instance to a file	CSV (comma separated values = COMPRESSED)  FWV (fixed width, columnar values = FORMATTED)  Compact (proprietary format)  SAP DB Page (proprietary format)	DATAEXTRACT, TABLEEXTRACT
Create a database		CATALOGLOAD

catalog or parts of a database catalog		
Unload a database catalog or parts of a database catalog		CATALOGEXTRACT
Migrate SAP DB instances between various versions and platforms		DBEXTRACT/DBLOAD



## Loading Data to Multiple Tables Simultaneously

### Syntax

`<if_condition> ::= IF <condition [Page 43]> | OTHERWISE`

### Use

This function is only supported by the [DATALOAD command \[Page 34\]](#).

If you want to use a **single** DATALOAD command to load data from one [source file \[Page 14\]](#) into multiple tables simultaneously, you must describe each of these [target tables \[Page 15\]](#) with a separate DATALOAD specification. Use selection criteria to define which records in the source file you want to load into which target tables.



You can only specify the **OTHERWISE** condition for the **last** table in a series of DATALOAD statements.

Unlike the usual DATALOAD procedure, a mass INSERT is not used for multiple target tables. Instead, the data records are transferred to the target tables individually from the data file. This affects the performance of the load procedure.

### Using IF <condition>



You want to load those data records in the `address.data` source file that begin with the letter **k** to the target table with the name `customer` and all those data records that begin with the letter **p** to the target table with the name `partner`:

```

DATALOAD TABLE customer      IF POS 1 = 'k'
  cno      2
  surname  3
  zip      5
  city     6
DATALOAD TABLE partner      IF POS 1 = 'p'
  cno      2
  surname  3
  zip      5
  city     6
INFILE 'address.data'
```

## Using IF <condition> | OTHERWISE



You want to load all data records in the `address.data` source file that are neither customer addresses nor partner addresses into the `prospects` target file.

```

DATALOAD TABLE customer      IF POS 1 = 'k'
  cno          2
  surname      3
  zip          5
  city         6
DATALOAD TABLE partner      IF POS 1 = 'p'
  cno          2
  surname      3
  zip          5
  city         6
DATALOAD TABLE prospects    OTHERWISE
  regno        2
  surname      3
  zip          5
  city         6
INFILE 'address.data'
```



## Loading Binary Values

### Use

All numeric [external data types \[Page 16\]](#) (INTEGER, REAL, DECIMAL, ZONED) are interpreted as binary values.

These can be converted to any of the [internal data types \[Page 16\]](#) (FIXED, SMALLINT, INTEGER, FLOAT) used by the database.

- Data of the data type INTEGER is always interpreted as signed values by the Replication Manager. Use the syntax rule [<int\\_spec> \[Page 71\]](#) to specify how these values are represented in the [source file \[Page 14\]](#).
- Data of the data type REAL is coded in host-specific format and cannot be transported between different operating systems and computer platforms without being converted again.

You can use the [FASTLOAD command \[Page 32\]](#) and the [DATALOAD command \[Page 34\]](#) to load external binary values into database tables.



```

DATALOAD TABLE hotel
  hnr          01-04 INTEGER
  name         09-18
  zip          20-25 DECIMAL
  city         27-36
  price        41-44 REAL
INFILE 'hotel.data' FORMATTED
```

### Prerequisites

You must specify the [source file \[Page 14\]](#) in the load command with either the [FORMATTED \[Page 19\]](#) format or the [FORMATTED BINARY \[Page 20\]](#) format. The file must be coded in binary.



## Selecting Data Records from the Source File

### Use

When you [load data \[Page 31\]](#), you can select the data records according to their contents.

The conditions defined here can be negated with **NOT**, linked with **AND** and **OR**, or encapsulated as required. The Replication Manager weights the operators accordingly ([condition \[Page 43\]](#)).



You want to load those data records from the [source file \[Page 14\]](#) `hotel.data` into the [target table \[Page 15\]](#) `hotel` that satisfy the following conditions:

The price in the data record is less than 400.00 (**IF**).

The name of the hotel is not City (**NOT**).

The hotel is located in BERLIN (**AND**).

The zip code is 13125 or 13126 (**OR**):

```

DATALOAD TABLE hotel
  IF POS 41-44 REAL < '400.00'
  AND
    POS 27-36 = 'BERLIN'
  AND
    (POS 20-25 = '13125' OR POS 20-25 = '13126')
  AND NOT
    POS 09-18 = 'City'
  hno      01-04 INTEGER
  name     09-18
  zip      20-25 DECIMAL
  place    27-36
  price    41-44 REAL
INFILE 'hotel.data' FORMATTED
  
```



You want to load those data records from the source file `hotel.data` into the target table `hotel` where the price is not more than 400,00:

```

DATALOAD TABLE hotel
  IF NOT POS 41-44 REAL > '400.00'
  hno      01-04 INTEGER
  name     09-18
  zip      20-25 DECIMAL
  place    27-36
  price    41-44 REAL
INFILE 'hotel.data' FORMATTED
  
```



## Terminology

[REPM Server \[Page 13\]](#)

[Command file \[Page 13\]](#)

[Source file \[Page 14\]](#)

[Target file \[Page 15\]](#)

[Source table \[Page 15\]](#)

[Target table \[Page 15\]](#)

[Qualification column \[Page 15\]](#)

[Remote server \[Page 15\]](#)

[Data types \[Page 15\]](#)

[Data file formats \[Page 18\]](#)

[Log file of the Replication Manager \[Page 21\]](#)



## REPM Server

User Manual: SAP DB → Terms → [REPM Server \[Extern\]](#)

The REPM Server must be installed on the host where the external data is located ([source files \[Page 14\]](#) or [target files \[Page 15\]](#)).



## Command File

When you [call the Replication Manager \[Page 21\]](#), you specify the name of the command file with the [option \[Page 22\]](#) →b. The commands in the command file are then send individually from the Replication Manager to the [REPM Server \[Page 13\]](#), where they are processed in this order.

Commands are used to formulate statements for loading and unloading data on the database instance or for changing the settings for the current user session. For example, you can change the SQL dialect or log on a new user so that data can be loaded to his or her tables. The external data files are also specified in these commands ([source files \[Page 14\]](#) or [target files \[Page 15\]](#)).

You can also use [SQL statements \[Page 84\]](#) in the command file.



```
repmcli -u jones,confidential -d MK1 -n PCnew -b command.dat
```



## Structure of a Command File

A command file can be built from the following components:

- [Keywords \[Page 84\]](#)
- [Commands for setting up a session \[Page 27\]](#) between the REPM Server and the database instance
- [Commands for loading and unloading data and catalog contents \[Page 31\]](#)
- [SQL statements \[Page 84\]](#) (creating tables, and so on)



## General Syntax Rules for Command Files

### Upper- and Lowercase Characters

A distinction is not made between upper- and lowercase characters. The Replication Manager converts all names to uppercase characters internally, unless you place character strings in double quotation marks.

## Comments

Comments in the [command file \[Page 13\]](#) are initiated by a slash at the start of the line. The entire line is then ignored when the [command file \[Page 13\]](#) is processed by the [REPM Server \[Page 13\]](#).

## Keywords

Both the [keywords \[Page 84\]](#) of the Replication Manager and SQL keywords can be used to denote database objects if they are placed in single or double quotation marks.

## Delimiters Between Commands

Individual commands in a command file are separated by means of a blank line or comment line.



```
CREATE TABLE customer
(cno          FIXED(4)
  CONSTRAINT cno BETWEEN 1 AND 9999   KEY,
 title        CHAR(7)
  CONSTRAINT title IN ('Mr', 'Ms', 'Company'),
 first name    CHAR(10),
 surname      CHAR(10) NOT NULL,
 zip          CHAR(5)  CONSTRAINT
  SUBSTR(plz_dom,1,5) LIKE '(0-9)(0-9)(0-9)(0-9)(0-9)',
 address      CHAR(25) NOT NULL)
/
FASTLOAD TABLE customer
  cno          1
  surname      2
  zip          3
  city         4
INFILE 'customer.data'
/
CREATE INDEX customer_index ON customer (surname)
/
DATAEXTRACT cno, surname, zip, city from customer
OUTFIELDS
  cno          1
  surname      2
  zip          3
  city         4
OUTFILE 'newcustomer.data'
```



## Source File

The term "source file" denotes the file from which data is loaded to the database instance using the Replication Manager.

The Replication Manager can process data stored in files, stored on tapes, or accessible with named pipes. It can also load data from data fields and/or data records in different [formats \[Page 18\]](#). You can specify these formats in the [command file \[Page 13\]](#) as a processing option for the source file. If you do not specify a format, the Replication Manager default (COMPRESSED) is used.

- [COMPRESSED \[Page 18\]](#)
- [FORMATTED \[Page 19\]](#)
- [FORMATTED BINARY \[Page 20\]](#)



## Target File

The term “target file” denotes the file to which the Replication Manager unloads data from a database instance.

The Replication Manager provides three [formats \[Page 18\]](#) for unloading the data. The default value is COMPRESSED.

- [COMPRESSED \[Page 18\]](#)
- [FORMATTED \[Page 19\]](#)
- [FORMATTED BINARY \[Page 20\]](#)



## Source Table

You can use the Replication Manager to load data from database instance tables to an external data file ([target file \[Page 15\]](#)).

Database tables from which data is loaded are referred to as source tables.



## Target Table

You can use the Replication Manager to load data from an external data file ([source file \[Page 14\]](#)) to the tables of a database instance.

Database tables to which data is loaded are referred to as target tables.



## Qualification Column

The values in the qualification column identify those lines in the [target table \[Page 15\]](#) that you want to update.



## Remote Server

The remote server is the node on which the [REPM Server \[Page 13\]](#) and the data are located.



## Data Types

With most [commands for loading and unloading data \[Page 31\]](#), you can specify the data type which the relevant field values have. This tells the [REPM Server \[Page 13\]](#) how to interpret the data in the data files.

When you load data, it is then converted into [internal database data types \[Page 16\]](#), according to these instructions, and loaded into the appropriate table columns. When you unload data, it is converted into external data types and unloaded from the appropriate table columns.



You cannot specify the data type with the [TABLELOAD \[Page 38\]](#) and [TABLEEXTRACT \[Page 40\]](#) commands.

The [external data type \[Page 16\]](#) and corresponding internal data type of the relevant column (column type) in the target table do not have to be identical.



The `article` database table, for example, can require the `FIXED` data type in the `ordered` column. As a result, data of the data type `INTEGER` is then converted to the `FIXED` data type.

```
FASTLOAD TABLE article
foa          01-08
des          09-39 CHAR
stock       40-43 INTEGER
min_ord     44-45 INTEGER
ordered     46-49 INTEGER
delivery_date 50-57 CHAR
price       58-65 DECIMAL(2)
weight      66-69 REAL
INFILE 'article.data' FORMATTED
```



## External Data Types

The term *external data type* denotes the [data type \[Page 15\]](#) of the external data in a [source file \[Page 14\]](#) or [target file \[Page 15\]](#).

A data type can be specified for a field in a [load or unload command \[Page 31\]](#) for each field value that needs to be loaded or unloaded (exception: `TABLEEXTRACT` and `TABLELOAD`).

If no data type is specified for a field value, the Replication Manager assumes the data type `CHARACTER` for this field value. If the field value does not have the `CHARACTER` data type, the data type must be specified explicitly.

The Replication Manager processes the following external data types:

[Plain text values \[Page 17\]](#)

CHAR	ASCII or EBCDIC coded, depending on the host, maximum of 254 bytes long
------	---

[Binary values \[Page 17\]](#)

INTEGER	Binary coded (host-specific), 1, 2 or 4 bytes long, sign in bit 0, negative values in two's complement notation
REAL	Floating point notation with mantissa and exponent, 4 or 8 bytes long, in host-specific notation
DECIMAL	Packed decimal: one digit per half byte, sign in extreme right half byte, 1 to 10 bytes long, maximum of 18 digits, the position of the decimal point is derived from the table column type
DECIMAL(n)	n specifies the number of digits on the right of the intended decimal point
ZONED	Zoned decimal: /370 zoned data format is permitted, the position of the decimal point is derived from the table column type
ZONED(n)	n specifies the number of digits on the right of the decimal point



## Data Types Used Internally by the Database

Internal data types are the [data types \[Page 15\]](#) used for the columns in the database table.





## Plain Text Values

Field values with the ASCII, EBCDIC, UCS2 or UTF8 code attribute are referred to in the following section as plain text values. They have the [data type \[Page 15\]](#) CHARACTER.

Plain text values can be transported between operating systems and computers without being converted first.

Plain text values can be used to input data into columns of any [internal data type used by the database \[Page 16\]](#) and can be converted to any data type. If you want to load plain text values into numerical columns, the values must be able to be interpreted as numbers.



## Binary Values

The following applies for the Replication Manager:

All numeric, external data types (INTEGER, REAL, DECIMAL, ZONED) are interpreted as binary values. These can be converted to any of the [internal data types \[Page 16\]](#) (FIXED, SMALLINT, INTEGER, FLOAT) used by the database.

- Data of the data type INTEGER is always interpreted as signed values by the Replication Manager. The way in which this data is represented in the data file can be adjusted with [<int spec> \[Page 71\]](#).
- Data of the data type REAL is coded in machine-specific format and cannot be transported between different operating systems and computer platforms without additional conversion.



## HEX Values

You can specify the external data type with a HEX value. This enables you to load or unload all data types in hexadecimal form. Every byte of data in the result is then represented by two hexadecimal numbers. Each value in the data file then occupies twice the amount of space as in the same format without the HEX specification.



```
FASTLOAD TABLE customer
  cno      01-03 CHAR HEX
  surname  04-12 CHAR HEX
  zip      13-17 INTEGER HEX
  place    18-29 CHAR HEX
INFILE 'customer.data' FORMATTED

DATALOAD TABLE article IF POS 40-47 INTEGER HEX > '0'
  foa      01-08
  des      09-39 NULL IF POS 09-11 = ' '
  stock    40-47 INTEGER HEX
  min_ord  47-48 INTEGER
  price    48-55 DECIMAL (2)
  weight   56-59 REAL
INFILE 'article.data' FORMATTED
```



## Data File Formats

Data files can have the following formats:

- [COMPRESSED \[Page 18\]](#)
- [FORMATTED \[Page 19\]](#)
- [FORMATTED BINARY \[Page 20\]](#)

You can specify the format for the data file in the [command for loading or unloading data \[Page 31\]](#).

The Replication Manager assumes that data files in [plain text \[Page 17\]](#) are in COMPRESSED format. If this is not the case, FORMATTED format or, with [binary \[Page 17\]](#) coded data files, FORMATTED BINARY format must be specified explicitly in the load or unload command.



## COMPRESSED

COMPRESSED specifies a specific [format for a data file \[Page 18\]](#).

If a [load or unload command \[Page 31\]](#) does not specify the format [FORMATTED \[Page 19\]](#) or [FORMATTED BINARY \[Page 20\]](#) explicitly, the system assumes that the data in the [source file \[Page 14\]](#) is in the COMPRESSED format (Replication Manager default). This means that it does not have to be specified explicitly in a load or unload command.

The COMPRESSED format can only be used for source files in [plain text \[Page 17\]](#). [Binary \[Page 17\]](#) data files must be loaded with the [FORMATTED BINARY \[Page 20\]](#) attribute.

If you do not specify a format in a load or unload command, however the data is still not in plain text, the Replication Manager creates an error message.



**customer.data**

Position number	1	2	3	4	....
	001,mueller,69185,Walldorf				
	002,schmidt,13403,Berlin				
	003,kleinert,25000,Upper-Lower				

The data fields do not have a standard format but are separated by commas:

FASTLOAD command:

```
FASTLOAD TABLE customer
  cno          1
  surname      2
  zip          3
  place        4
INFILE 'customer.data'
```

## Data Lines

In COMPRESSED format, a data line must at least be long enough to represent the data. The length of the individual data fields and the total length of the data lines can vary. Each data line ends with a line break.

When data is loaded, a data line in a [source file \[Page 14\]](#) in this format corresponds to exactly one data record in the [target table \[Page 15\]](#).

When data is unloaded, a data line in a [source table \[Page 15\]](#) in this format corresponds to exactly one data record in the [target file \[Page 15\]](#).

## Data Fields

Data fields in the data files must have the data type CHAR.

The individual data fields are separated by a freely selectable character (default value of the Replication Manager: comma). They can also be encapsulated in freely selectable characters (default value of the Replication Manager: double quotation marks).

If delimiters follow each other directly in a data line, then the value is the empty character string. If the relative position is larger than the number of values in the line, the value is also the empty character string.

## Specified Position

The assignment of a data file data field to a column in the table is determined by the position [<field\\_pos> \[Page 45\]](#) in a [command for loading or unloading data \[Page 31\]](#). This specifies a relative position within a data record. In other words, it specifies the value in a data record that is to be loaded into or unloaded from a specific column.

When you load data from a source table, the data fields in the target table in the database must be sorted in ascending order. You can also assign the same input values to different columns or even omit positions. If consecutive delimiters are detected or if the relative position is greater than the number of values in the line, the value is the empty character string.

When you unload data from the database into a target file, assign the position 1 to the first column in your list. The position numbers of the following columns rise by 1 each time. This also means that each position number can only be assigned once.



## FORMATTED

FORMATTED specifies a [format for a data file \[Page 18\]](#). Since this format is not the Replication Manager default, you must specify it explicitly in a [command for loading or unloading data \[Page 31\]](#).

The FORMATTED format can only be used for [source files \[Page 14\]](#) in [plain text \[Page 17\]](#). [Binary \[Page 17\]](#) data files must be loaded with the [FORMATTED BINARY \[Page 20\]](#) attribute.

## Data Lines

A data line in a data file corresponds to a data record that you want to load or unload.

All data lines have the same fixed length. Each line in the data file contains the individual data fields at the same position with a fixed length. Each data line ends with a line break.

## Data Fields

The assignment between data fields in the data file and the table columns is made according to the position in a command for loading or unloading data.

## Specified Position

The data field in the data file are described by their byte start and end position. The first possible byte start position is 1. Specifying an end position is optional. If no end position is specified for a data field, it has a length of 1.

The assignment between the data fields in the data file and table columns can be made in any order. The order does not affect the processing speed.

The positions do not need to follow each other directly when you load or unload data. When it unloads data, the Replication Manager fills any gaps with blank characters.



customer.data

Position no.	1 2 3 4 5 6 7 8 9 .....
	0 1 m u e l l e r - 6 9 1 8 5 W a l l d o r f - - - 0 2 s c h m i d t - 1 3 4 0 3 B e r l i n - - - - - 0 3 k l e i n e r t 2 5 0 0 0 U p p e r - L o w e r

The data fields have a standard format.

FASTLOAD command:

```
FASTLOAD TABLE customer
  cno          01-02
  surname      03-10
  zip          11-15
  place        16-27
INFILE 'customer.data' FORMATTED
```



## FORMATTED BINARY

FORMATTED BINARY specifies a [format for a data file \[Page 18\]](#). Since this format is not the Replication Manager default, you must specify it explicitly in a [command for loading or unloading data \[Page 31\]](#).

This format can be used to load and unload [binary values \[Page 17\]](#).

The FORMATTED BINARY format is largely the same as the [FORMATTED \[Page 19\]](#) format. The only difference is that the length of a data record is only determined by the total length of the individual data fields.

## Data Fields

The data fields can contain special characters and all have the same fixed length. This also applies to the individual data records.

The assignment between data fields in the data file and the table columns is made according to the [<field\\_pos> \[Page 45\]](#) position in the command for loading or unloading data.

Unlike the FORMATTED format, a data record does **not** end with an additional line break.

## Specified Positions

Data fields are described by their byte start and end position. The first possible byte start position of a data record is 1. Specifying an end position is optional. If you do not specify an end position, the corresponding data field has a length of 1.

The data fields of the data file can be assigned in any order to the table columns. The order does not affect the processing speed.

The positions do not need to follow each other directly when you load or unload data. When it unloads data, the Replication Manager fills any gaps with blank characters.



customer.data

Position no.	1 2 3 4 5 6 7 8 9 .....
	0 1 m u e l l e r x 6 9 1 8 5 W a l l d o r f                      0 2 s c h m i d t                      1 3 4 0 3 B e r l i n                      0 3 k l e i n e r t

	2 5 0 0 0 U p p e r - L o w e r
--	---------------------------------

The data fields have a standard format.

FASTLOAD command:

```
FASTLOAD TABLE customer
  cno          01-02
  surname      03-10
  zip          11-15
  place        16-26
INFILE 'customer.data' FORMATTED BINARY
```



## Log File of the Replication Manager

When the [REPM Server \[Page 13\]](#) is started, it creates or writes to a log file. The Replication Manager defaults for the name of this file is `repserver.log`. The file is located in the `wrk` directory of a SAP DB installation.

The REPM Server logs the following information to this file:

- Database name and user name
- Start and end of a load or unload action
- All executed SQL statements
- All executed load commands
  - For each load command, the position of the rejected input records in the data file and the cause of error
  - For each load command, the accumulated counts (inserted/modified lines and rejected lines)
- All executed unload commands
  - For each DATAEXTRACT command, the accumulated counts (number of extracted lines, and the number of lines extracted with errors)
- A message if the action was terminated

The log file is structured in such a way that parts of it can be easily used to create a command file. All comments in a new record start with `/ *` or `/ <letter>` and are ignored by the [REPM Server \[Page 13\]](#).

All message or error text lines in the log are identified by a letter at the start of the line.

/ M	Stands for system messages, such as the number of lines inserted and rejected, last completed transaction, and so on.
/ E	Line with error messages This is used to introduce the two lines with the table and position values as well as the error number or error text for rejected lines.



## Calling the Replication Manager

- Using the REPMCLI
- Using script languages

The [REPM Server \[Page 13\]](#) uses [options \[Page 22\]](#) and [commands \[Page 26\]](#).

Options are transferred to the Replication Manager when it is called.

Commands are transferred from the Replication Manager to the REPM Server using a [command file \[Page 13\]](#).

When a connection is set up between the REPM Server and the database instance, the Replication Manager uses the specified options first. Following this, any [commands for setting up the connection \[Page 27\]](#) contained in the command file are evaluated and the [commands for loading or unloading data \[Page 31\]](#) are executed.

## Options

The Replication Manager uses the following options ([Calling the Replication Manager \[Page 21\]](#)):

<a href="#">-b &lt;command_file&gt; [Page 22]</a>
<a href="#">-d &lt;database_name&gt; [Page 23]</a>
<a href="#">-E &lt;number&gt; [Page 23]</a>
<a href="#">-h [Page 23]</a>
<a href="#">-n &lt;server_node&gt; [Page 24]</a>
<a href="#">-o[w a] &lt;file_name&gt; [Page 24]</a>
<a href="#">-p &lt;number&gt;&lt;substitution_string&gt; [Page 24]</a>
<a href="#">-R &lt;directory_name&gt; [Page 25]</a>
<a href="#">-r &lt;server_node&gt; [Page 25]</a>
<a href="#">-u &lt;userid&gt;,&lt;password&gt; [Page 26]</a>
<a href="#">-V [Page 26]</a>
<a href="#">-v [Page 26]</a>

## Syntax

```
repcli [<options>] -b <command_file>
```



```
repcli -u jones,confidential -d MK1 -b command.dat
```

## Use

When you call up the Replication Manager and specify a user and the name of the database instance (options `-u` and `-d`), you establish the connection between the [REPM Server \[Page 13\]](#) and the specified database instance for this user. All of the commands are then transferred in the form of a [command file \[Page 13\]](#) (option `-b`).

You can also call up the Replication Manager first and then establish the connection between the REPM Server and the database instance by transferring [commands for setting up a session \[Page 27\]](#) in a command file.



## Name of Command File: -b

### Syntax

```
-b <command_file>
```

<code>&lt;command_file&gt;</code>	Command File
-----------------------------------	--------------

## Use

[Option \[Page 21\]](#) for calling the Replication Manager when a connection is set up between the [REPM Server \[Page 13\]](#) and the database instance.

The contents of the [command file \[Page 13\]](#) are transferred to the REPM Server.



## Name of the Database Instance: -d

### Syntax

-d <database\_name>

<database_name>	Name of the database instance, all <a href="#">REPM Server commands [Page 26]</a> then refer to this database instance.
-----------------	---

## Use

[Option \[Page 22\]](#) for calling the Replication Manager when a connection is set up between the [REPM Server \[Page 13\]](#) and the database instance.



The name of the database instance can also be transferred with the [USE SERVERDB \[Page 29\]](#) or [USE USER \[Page 30\]](#) command in the [command file \[Page 13\]](#), and can therefore be changed during the user session with the Replication Manager.



## Maximum Permitted Number of Errors: -E

### Syntax

-E <number>

<number>	Maximum permitted number of commands with errors
----------	--

## Use

You use this [option \[Page 22\]](#) to specify after how many commands with errors the Replication Manager stops processing the [command file \[Page 13\]](#).



## Help for the Options: -h

### Syntax

-h

## Use

[Option \[Page 22\]](#) for calling the Replication Manager when a connection is set up between the [REPM Server \[Page 13\]](#) and the database instance.

Help is displayed for the available options.



## Name of the Host Node of the Database Instance: -n

### Syntax

-n <server\_node>

<server_node>	Name of the host node of the database instance, Replication Manager default is the name of the local host
---------------	---

### Use

[Option \[Page 22\]](#) for calling the Replication Manager when a connection is set up between the [REPM Server \[Page 13\]](#) and the database instance.



The host node name can also be transferred with the [USE SERVERDB \[Page 29\]](#) or [USE USER \[Page 30\]](#) commands in the [command file \[Page 13\]](#).



## Output File of the Replication Manager: -o[w|a]

### Syntax

-o[w|a] <file\_name>

<file_name>	Name of the file in which the Replication Manager output is written. The default for all output is the screen.
-------------	---

### Use

[Option \[Page 22\]](#) for calling the Replication Manager when a connection is set up between the [REPM Server \[Page 13\]](#) and the database instance.

#### Option -ow

The file is opened and any existing entries are deleted.

#### Option -oa

The file is opened and the outputs are appended to the end of the file without any existing entries being deleted.



This option exclusively refers to outputs from the Replication Manager program. The program still writes a [log file \[Page 21\]](#) irrespectively.



## Substitution Parameter: -p

### Syntax

-p<number> <substitution\_value>

<number>	Defines the number n of the placeholder you want to substitute in the command file, where $9 \leq n \leq 1$ .
<substitution_value>	Value that substitutes the placeholder in the command file



## Use

Use this [option \[Page 22\]](#) to determine values for placeholders in the command file. These values can be any character strings. You can specify 9 different parameters (and placeholders) for each command file.

In the command file, enter the placeholder in the form `&<number>`.



Entry in the command file:

```
DATAEXTRACT FOR DATALOAD TABLE customer OUTFILE &1 OUTFILE
'customer.data'
DATAEXTRACT FOR FASTLOAD TABLE article OUTFILE &1 APPEND OUTFILE
'article.data'
```

As an option when you call the Replication Manager:

```
repmcli ... -p1 'load.command'
```



## InstRoot Directory: -R

### Syntax

`-R <directory_name>`

`<directory_name>`

Name of the InstRoot directory on the host node of the REPM server  
The Replication Manager does not have a default.

### Use

[Option \[Page 22\]](#) for calling the Replication Manager when a connection is set up between the [REPM server \[Page 13\]](#) and the Replication Manager.

If multiple versions of the database software are installed on the host, you can start a specific version of the REPM server. If you do not specify a directory name, the REPM server for the newest version of the database software is started.



## Name of the REPM Server Host Node: -r

### Syntax

`-r <server_node>`

`<server_node>`

Node name of the host where the [REPM Server \[Page 13\]](#) is located;  
Replication Manager default is the name of the local host

### Use

[Option \[Page 22\]](#) for calling the Replication Manager when a connection is set up between the REPM Server and the Replication Manager client (REPMCLI).



## User Data: -u

### Syntax

-u <userid>,<password>

<userid>	User name
<password>	User password

### Use

[Option \[Page 21\]](#) for calling the Replication Manager when a connection is set up between the [REPM Server \[Page 13\]](#) and the database instance.

A user session for the specified user is set up with the database kernel.



User data can also be transferred with the [USE USER \[Page 30\]](#) command in the [command file \[Page 13\]](#).



## REPM Server Version: -V

### Syntax

-v

### Use

If you specify this [option \[Page 22\]](#) when a connection is set up between the [REPM Server \[Page 13\]](#) and the database instance, the version of the REPM Server appears.



## Version of the Replication Manager: -v

### Syntax

-v

### Use

Use this [option \[Page 22\]](#) to find out the version of the Replication Manager program.



## REPM Server Commands

Each Replication Manager command comprises one or more keywords, arguments, and options ([Calling the Replication Manager \[Page 21\]](#)).

[Commands for Setting Up a Session \[Page 27\]](#)

[Commands for Loading and Unloading Data \[Page 31\]](#)

[Syntax Rules for Commands \[Page 41\]](#)

[SQL Statements \[Page 84\]](#)

[Keywords and Alternative Keywords \[Page 84\]](#)

[Processing Commands and Statements \[Page 88\]](#)



## Commands for Setting Up a Session

The Replication Manager uses these commands to set up a connection between the [REPM Server \[Page 13\]](#) and the database kernel. Commands are transferred from the Replication Manager to the REPM Server in the form of a [command file \[Page 13\]](#), which is entered when you [call the Replication Manager \[Page 21\]](#).

The following commands can be used in a command file to set up a session:

[AUTOCOMMIT command \[Page 27\]](#)

[IGNORE TERMCHARSET command \[Page 28\]](#)

[SET command \[Page 28\]](#)

[SQLMODE command \[Page 29\]](#)

[USE SERVERDB command \[Page 29\]](#)

[USE TERMCHARSET command \[Page 29\]](#)

[USE USER command \[Page 30\]](#)

[USE USERKEY command \[Page 30\]](#)

[VERSION command \[Page 30\]](#)



## AUTOCOMMIT Command

### Syntax

```
<autocommit_statement> :: = AUTOCOMMIT ON | AUTOCOMMIT OFF
```

### Use

You use this [command for setting up a session \[Page 27\]](#) to activate/deactivate AUTOCOMMIT mode for the database session.

### AUTOCOMMIT ON

AUTOCOMMIT mode is activated by default by the Replication Manager.

With [commands for loading data \[Page 31\]](#), a COMMIT is used to complete the insertion of a certain number of data records. The user can determine this number by means of the REPM Server command SET TRANSACTION SIZE. This mode is not relevant when data is unloaded, because the table contents in the database are not modified.

If a [command file \[Page 13\]](#) contains both COMMIT and ROLLBACK statements when AUTOCOMMIT mode is active, they will be ignored by the Replication Manager.

Each [SQL statement \[Page 84\]](#) is terminated implicitly by the [REPM Server \[Page 13\]](#) using a COMMIT.

### AUTOCOMMIT OFF

AUTOCOMMIT mode is deactivated.

All transactions must be terminated explicitly by means of a COMMIT in the command file.

You can group commands in units so that they can be completed or reset at once.

This mode applies to all SQL statements and [REPM Server commands \[Page 26\]](#), with the exception of [FASTLOAD \[Page 32\]](#), [TABLEEXTRACT \[Page 40\]](#) and [TABLELOAD \[Page 38\]](#). These commands

run outside of the database transaction concept and the REPM Server always terminates them implicitly with COMMIT.



Large transactions (a large number of data records to be loaded, for example) need a correspondingly large database log.

## IGNORE TERMCHARSET

### Syntax

```
<ignore_termcharset_statement> ::= IGNORE TERMCHARSET
```

### Use

This [command for setting up a session \[Page 27\]](#) ends the current connection between the [REPM Server \[Page 13\]](#) and the database instance and sets up a new connection with the logon data of the previous connection, that is, it copies the user name, password, name of the database instance, and host node name. It does not use a character conversion table.

## SET Command

### Syntax

```
<set_statement> ::= SET CODETYPE <standard_code_spec> [Page 81];
| SET DATE <standard_date_mask> [Page 81];
| SET TIME <standard_time_mask> [Page 82];
| SET TIMESTAMP <standard_timestamp_mask> [Page 82];
| SET <null_spec> [Page 80];
| SET <bool_spec> [Page 68];
| SET <number_spec> [Page 79];
| SET LANGUAGE ENG|DEU;
| SET MAXERRORCOUNT <valMAXERRORCOUNT>
| SET COMPRESSED /<s>/<d>/;
```

<b>LANGUAGE ENG DEU</b>	Language in which the messages of the database instance and the REPM Server appear <b>ENG</b> : English, <b>DEU</b> : German
<b>COMPRESSED /&lt;s&gt;/&lt;d&gt;/</b>	Characters used in the COMPRESSED file format to separate or select data; current values can be changed for individual commands  <b>&lt;s&gt;</b> Character used to separate data fields, default value: comma, value must be exactly one character long, and cannot be blank  <b>&lt;d&gt;</b> Character used to select data, default value: double quotation marks, value can be exactly one character long or blank, if it is blank, no delimiter is defined
<b>MAXERRORCOUNT &lt;valMAXERRORCOUNT&gt;</b>	Definition of how many errors the Replication Manager accepts during processing of an individual command before processing of the command is terminated This value applies during a complete session or until it is overwritten by another SET MAXERRORCOUNT command

## Use

You can use this [command for setting up a session \[Page 27\]](#) to adapt the default values in the control parameters of the [REPM Server \[Page 13\]](#) to your own needs.



The new values are only valid for the current session and must be restored if they are needed in subsequent sessions.

## SQLMODE Command

### Syntax

```
<sql_mode_statement> ::= SQLMODE <sql_mode> [Page 42];
```

### Use

The [commands for loading and unloading data \[Page 13\]](#) and [SQL statements \[Page 31\]](#) contained in a [command file \[Page 84\]](#) are interpreted in accordance with the defined SQL mode.

This [command for setting up a session \[Page 27\]](#) switches the SQL mode on the database. The current mode remains active until it is changed again with this command. If you enter an invalid SQL mode, the Replication Manager uses the internal SQL mode of the database instance.

## USE SERVERDB Command

### Syntax

```
<use_serverdb_statement> ::= USE <database_name_statement> [Page 41]
```

### Prerequisites

A user with the same name and password must exist on the new database instance.

### Use

Use this [command for setting up a session \[Page 27\]](#) to switch to another database instance without exiting the Replication Manager. It terminates the connection to the current database instance and sets up the new connection to the database instance.

All subsequent commands are referred to this database instance by the Replication Manager.



```
USE SERVERDB MK1 ON PCnew;
```

## USE TERMCHARSET Command

### Syntax

```
<use_termcharset_statement> ::= USE TERMCHARSET <valCHARSET_NAME>
```

<valCHARSET_NAME>	Name of the new character conversion table If a distinction has to be made between upper and lower-case letters in this name, simply place it in single or double quotation
-------------------	--

	marks.
--	--------

## Use

This [command for setting up a session \[Page 27\]](#) ends the current connection between the [REPM Server \[Page 13\]](#) and the database instance and sets up a new connection with the logon data of the previous connection, that is, it copies the user name, password, name of the database instance, and host node name. A different character conversion table, however, is used.

## USE USER Command

### Syntax

```
<use_user_statement> ::= USE <user_statement> [Page 42];
```

## Use

This [command for setting up a session \[Page 27\]](#) terminates the current connection of the [REPM Server \[Page 13\]](#) to the database instance and sets up a new connection with the specified parameters.

If you do not enter a database instance under <user\_statement>, the new connection is created to the database instance that was used prior to the USE USER command.

## USE USERKEY Command

### Syntax

```
<use_userkey_statement> ::= USE USERKEY <valUSERKEY>;
```

## Use

The [command for setting up a session \[Page 27\]](#) USE USERKEY is a variant of the [USE USER command \[Page 30\]](#). An XUSER entry (with the key specified in valUSERKEY) is used here to provide all of the required information about the new user.

**See also:** *User Manual: SAP DB*

## VERSION Command

### Syntax

```
<version_statement> ::= VERSION;
```

## Use

You use this [command for setting up a session \[Page 27\]](#) you instruct the REPM Server to write information relating to the version of the server to the [log file \[Page 21\]](#).



```
VERSION = 7.2.1
BUILD   = REPMServer 7.2.1      Build 002-000-112-746
OS      = WIN32
```

CODE = ASCII  
SWAP = full



## Commands for Loading and Unloading Data and Catalog Contents

### Command Structure

Commands for loading and unloading data and catalog contents can be entered in the [command file \[Page 13\]](#), which is entered when you [call the Replication Manager \[Page 21\]](#). The commands consist of two sections:

	Section 1	Section 2
<b>Load commands</b>	<ul style="list-style-type: none"> <li>• <a href="#">Keyword [Page 84]</a> for the load operation (for example, <b>FASTLOAD</b>)</li> <li>• Description of the <a href="#">target table [Page 15]</a> structure and the assigned of data fields in the <a href="#">source file [Page 14]</a> to the columns in the target table</li> </ul>	<ul style="list-style-type: none"> <li>• Keyword <b>INFILE</b></li> <li>• Unchanged name of source file</li> </ul>
<b>Unload commands</b>	<ul style="list-style-type: none"> <li>• Keyword for the unload operation (for example, <b>DATAEXTRACT</b>)</li> <li>• Database query to be used to extract the data from the <a href="#">source table [Page 15]</a></li> </ul>	<ul style="list-style-type: none"> <li>• Keyword <b>OUTFILE</b></li> <li>• Unchanged name of <a href="#">target file [Page 15]</a></li> </ul>

### Overview

#### [FASTLOAD command \[Page 32\]](#)

**FASTLOAD** [[<usage\\_spec> \[Page 49\]](#)] **TABLE** [<table\\_spec> \[Page 49\]](#)  
[<load\\_column\\_spec\\_mlt> \[Page 58\]](#) [<infile\\_spec> \[Page 71\]](#)

#### [DATALOAD command \[Page 34\]](#)

**DATALOAD TABLE** [<table\\_spec> \[Page 49\]](#) [[<duplicates\\_clause> \[Page 44\]](#)]  
[<load\\_column\\_spec\\_mlt> \[Page 58\]](#) [<infile\\_spec> \[Page 71\]](#) [[<longfile\\_spec\\_mlt> \[Page 78\]](#)]

#### [DATAEXTRACT command \[Page 36\]](#)

**DATAEXTRACT** [<select\\_expression> \[Page 65\]](#) [**OUTFIELDS** [<output\\_column\\_list> \[Page 63\]](#)] [<extract\\_files\\_spec> \[Page 69\]](#)  
| **DATAEXTRACT** [<restore\\_spec> \[Page 47\]](#) **TABLE** [<table\\_name> \[Page 48\]](#)  
[[<order\\_clause> \[Page 62\]](#)] [<extract\\_files\\_spec> \[Page 69\]](#)

#### [DATAUPDATE command \[Page 37\]](#)

**DATAUPDATE TABLE** [<table\\_spec> \[Page 49\]](#) [<acc\\_column\\_spec\\_mlt> \[Page 51\]](#)  
[<set\\_column\\_spec\\_mlt> \[Page 66\]](#) [<infile\\_spec> \[Page 71\]](#) [[<longfile\\_spec\\_mlt> \[Page 78\]](#)>]

[TABLELOAD command \[Page 38\]](#)

TABLELOAD [<part\\_spec> \[Page 80\]](#) [<infile\\_spec> \[Page 71\]](#) [<restart\\_outfile\\_spec> \[Page 80\]](#) > [RESTART]

[TABLEEXTRACT command \[Page 40\]](#)

TABLEEXTRACT [<part\\_spec> \[Page 80\]](#) [<outfile\\_spec> \[Page 80\]](#)

CATALOGLOAD

CATALOGLOAD

CATALOGEXTRACT

CATALOGEXTRACT

DBLOAD

DBLOAD

DBEXTRACT

DBEXTRACT



```
FASTLOAD TABLE customer
  cno          1
  surname      2
  zip          3
  city         4
INFILE 'customer.data'
```



```
DATAEXTRACT cno, surname, zip, city from customer OUTFIELDS
  cno          1
  surname      2
  zip          3
  city         4
OUTFILE 'newcustomer.data'
```



## FASTLOAD Command

### Syntax

<fastload\_statement> ::=

FASTLOAD [[<usage\\_spec> \[Page 49\]](#)] TABLE [<table\\_spec> \[Page 49\]](#) [<load\\_column\\_spec\\_mlt> \[Page 58\]](#) [<infile\\_spec> \[Page 71\]](#)



```
FASTLOAD with 100 % USAGE
TABLE customer
  cno          1
  surname      2
```



```
zip          3
place       4
INFILE 'customer.data'
```

## Use

You use this [command for loading data \[Page 31\]](#) to load external data into the tables of a database instance.

FASTLOAD command files are processed while the database is running.



When data is loaded with FASTLOAD, it is entered directly in the pages on the database instance. This command loads data more quickly than the [DATALOAD command \[Page 34\]](#). However, because it does not write any log entries, you must back up the new data after you have loaded it. You can back up either the relevant pages (incremental data backup) or the entire data volume. The table is write-protected until you have backed up the new data. **See:** *Database Manager GUI: SAP DB 7.3, section Data Backup*

## Prerequisites

- The target table exists on the database instance. Otherwise, you can create a target table with a suitable [SQL statement \[Page 84\]](#) before the FASTLOAD command in the command file.
- You have logged onto the Replication Manager with the user who is the owner of the target table.
- The data in the source file is sorted in ascending order in the sequence of the primary key.
- The target table does not have an index.
- If the target table already contains data, only those data records whose key values are greater than the largest key value in the target table can be inserted.
- The data that you want to load does not contain LONG columns.

## Process Flow

When a FASTLOAD command is started, the target table of the load operation in the database instance is locked so other users cannot write to it. The target table can still be read, however.

Unrestricted reading and writing is possible in all other tables.

Once a FASTLOAD command has been processed, other write operations triggered with FASTLOAD from the same user can be performed on this table. Once the load operation with FASTLOAD has been completed, all users only have read access for the table.

A backup of the database instance must be created before it is released again for write operations from other users.

## Result

The data in the [source table \[Page 15\]](#) has been loaded to the target table.



The [REPM Server \[Page 13\]](#) uses an implicit COMMIT to complete the data load successfully.

## Errors

If a FASTLOAD load operation is aborted, all of the lines transferred during the operation are deleted. Data that existed in the table before the FASTLOAD load operation was started remains unchanged.

If you cancel the load operation because the Replication Manager refused to transfer certain lines, you can check the [log file of the Replication Manager \[Page 21\]](#) to find the data record that caused the operation to fail.

## DATALOAD Command

### Syntax

`<dataload_statement> ::=`

`DATALOAD TABLE <table_spec> [Page 49] [<duplicates_clause> [Page 44]]`  
`<load_column_spec_mlt> [Page 58] <infile_spec> [Page 71] [<longfile_spec_mlt> [Page 78]]`



```
DATALOAD TABLE customer
  cno          01-04
  surname      06-12
  zip          14-18
  city         20-31
INFILE 'customer.data' FORMATTED
```



```
DATALOAD TABLE article
  foa          01-08 CHAR
  des          09-39 CHAR
  stock        40-43 INTEGER
  min_ord      44-45 INTEGER
  ordered      46-49 INTEGER
  del_date     50-57 CHAR
  price        58-65 DECIMAL (2)
  weight       66-69 REAL
INFILE 'article.data' FORMATTED
```

[Further examples \[Page 35\]](#)

### Use

The DATALOAD command is a [command for loading data \[Page 31\]](#). Like the [FASTLOAD command \[Page 32\]](#), the DATALOAD command reads data from a file and writes it to a database instance table.

DATALOAD [command files \[Page 13\]](#) are processed while the database is in operation.



Backing up the log at the same time can impair performance with the result that the command file takes longer to process than with FASTLOAD. Writing the log simultaneously, however, allows the entries to be undone if the event of an error by reviewing the log.

### Prerequisites

The target table exists on the database instance. Otherwise, it must be created before the DATALOAD command is executed.

### Process Flow

The Replication Manager generates an internal INSERT command from the DATALOAD command and then executes it. The [REPM Server \[Page 13\]](#) uses a communication packet to transfer the data records directly to the relevant table on the database instance. Log entries are written for these table changes.

During the load operation, all of the tables modified by this DATALOAD command can also be read and changed by other users.

## Result

The data in the [source file \[Page 14\]](#) has been loaded into the [target table\(s\) \[Page 15\]](#). All of the changes made to the target table(s) as a result have also been written to the log on the database instance.

## Errors

If the load operation cannot be ended successfully, the Replication Manager logs the last row that was entered successfully in the data file, the number of rows inserted, and the number of rows rejected ([Replication Manager log file \[Page 21\]](#)).



## Examples of DATALOAD Commands

```
DATALOAD TABLE booking IF POS 4 >= '11/08/1999"
  pno      1
  cno      2
  hno      3
  from_dat 4
  to_dat   5
  price    6
INFILE 'master.data'
  DATE 'mm/dd/yyyy'

DATALOAD TABLE hotel
  IF (POS 41-44 REAL < '1000.00')
  AND
    (POS 41-44 REAL >= '1000.00')
    hno      01-04 INTEGER
    name     09-18
    zip      20-25 DECIMAL
    place    27-36
    price    41-44 REAL
INFILE 'hotel.data' FORMATTED

DATALOAD TABLE article
  foa      01-08
  des      09-39          NULL IF POS 09-11 = ' '
  stock    40-43 INTEGER NULL IF POS 40-43 INTEGER < '0'
  min_ord  44-45 INTEGER
  price    46-53 DECIMAL (2) NULL IF POS 1 <> 'X'
                                OR POS 46-53 DECIMAL < '0'
  weight   54-57 REAL
INFILE 'article.data' FORMATTED

DATALOAD TABLE address
  id      1
  telephone 2 DEFAULT NULL
  place    3 DEFAULT NULL
INFILE 'customer.data'
NULL '-'

DATALOAD TABLE article
  changed_by USER
  changed_on DATE
  changed_at TIME
  foa      1
  des      2
  stock    3 NULL IF POS 3 < '0'
INFILE 'customer.data'
```

## DATAEXTRACT Command

### Syntax

```
<dataextract_statement> ::=  
DATAEXTRACT <select_expression> [Page 65] [OUTFIELDS] [<output_column_list> [Page 63]]  
<extract_files_spec> [Page 69]  
| DATAEXTRACT <restore_spec> [Page 47] TABLE <table_name> [Page 48] [<order_clause>  
[Page 62]] <extract_files_spec> [Page 69]
```

### Use

This [command for unloading data \[Page 31\]](#) unloads data from database tables into external files ([target files \[Page 15\]](#)). You can also define the format of the target file ([data file formats \[Page 18\]](#)).

You can use the [DATALOAD \[Page 34\]](#) or [FASTLOAD \[Page 32\]](#) commands to reload data that has been unloaded into external data files back into the database.

The Replication Manager has two variants of the DATAEXTRACT command.

### Prerequisite

You must have access authorization for the tables from which you want to unload data.

You must be owner of the table to use the second variant of the command.

### DATAEXTRACT Command: Variant 1

You can do the following with this variant:

- Specify which columns of a [source table \[Page 15\]](#) you want to unload
- Link multiple tables that you want to unload
- Specify a sort sequence for the extracted data



```
DATAEXTRACT cno, surname, zip, location from customer
```

```

OUTFILE customer.data'

DATAEXTRACT * from article
OUTFIELDS  foa          01-08
           des          09-39
           stock        40-43 INTEGER
           min_ord      44-45 INTEGER
           ordered      46-49 INTEGER
           del_date     50-57
           price        58-65 DECIMAL (2)
           weight       66-69 REAL
OUTFILE 'article.data' FORMATTED

```

## DATAEXTRACT Command: Variant 2

Unlike the first variant, you can only use variant 2 to **completely unload individual tables**. You unload all data from a table into a target file, at the same time generating a [command file \[Page 13\]](#), which you can use to recover the unloaded table completely.

This command file contains:

- A DATALOAD, FASTLOAD or [DATAUPDATE command \[Page 37\]](#)

If the unloaded table is a Basis table ([see also Reference Manual 7.2](#), section Concepts, Table) and the user is the owner of the table, the command file also contains a

- CREATE-TABLE statement

```

DATAEXTRACT FOR DATALOAD TABLE article
OUTFILE 'article.command'
OUTFILE 'article.data'

```

## Process Flow

The Replication Manager uses the information from the DATAEXTRACT command to generate an internal SELECT command and constructs a results table. The data in this results table is then unloaded into the target file, as instructed by the command.

All tables from which data is being unloaded are write-protected while the DATAEXTRACT command is being executed. This means that other users cannot make changes to this table while it is being unloaded.

## Result

The data is unloaded from the tables into the target file.

## Errors

If the unload command could not be completed successfully, the Replication Manager logs the number of successfully unloaded data records ([Replication Manager log file \[Page 21\]](#)).

## DATAUPDATE Command

### Syntax

```

<dataupdate_statement> ::= DATAUPDATE TABLE <table_spec [Page 49]>
<acc_column_spec_mlt [Page 51]> <set_column_spec_mlt [Page 66]> <infile_spec [Page 71]>
[<longfile_spec_mlt [Page 78]>]

```

## Use

This [command for loading data \[Page 31\]](#) uses data in the source file to update individual column values in lines in database tables (target tables).

## Prerequisites

The target table exists on the database instance.

## Process Flow

The Replication Manager generates an internal UPDATE command from the DATAUPDATE command and then executes it.

## Result

The individual line values in the specified source table columns are updated. All of the changes made to the target table(s) as a result have also been written to the log on the database instance.

## Errors

If errors occur while a DATAUPDATE command is being executed, the Replication Manager terminates the command and displays an error message.

## TABLELOAD Command

### Syntax

```
<tableload_statement> ::= TABLELOAD <part\_spec> \[Page 80\] <infile\_spec> \[Page 71\]  
<restart_outfile_spec \[Page 80\]> [RESTART]
```



Use the syntax rule `restart_outfile_spec` to specify the restart file. The Replication Manager stores the table name of the last successfully loaded table in this file to enable a restart after an error. The restart file must **always** be specified.



```
TABLELOAD TABLE customer INFILE 'customer.data' outfile  
'restart.dat'
```

## Use

Use this [command for loading data \[Page 31\]](#) to recover complete database tables ([target tables \[Page 15\]](#)), that is schema and data, from the data in external files ([source files \[Page 14\]](#)).

The source file format is specified by the internal database format. You cannot change it.

The table that you want to recover does not need to exist in the database instance.

This command does not write any log entries.

## Prerequisites

The [TABLEEXTRACT command \[Page 40\]](#) created the source file.

## Use

### Case 1:

#### The table exists in the database instance

The REPM Server can adjust the following differences in schema between the source tables and target tables.

- Table name and column name of source table and target table  
The table name and column names of the source table are copied.
- Differently defined defaults  
The defaults defined in the source table are copied. The defaults defined in the target table are deleted.
- Differently defined indexes  
The indexes defined in the source table are copied. The indexes defined in the target table are deleted.
- Differently defined constraints  
The constraints defined in the source table are copied. The constraints defined in the target table are deleted.

The REPM Server does **not** adjust the following differences in schema:

- Type and length of individual columns  
The entire schema of the target table is replaced by the schema of the source table.

### Case 2:

#### The table does not exist in the database instance

A new table is created in the database instance.

## Process Flow

The Replication Manager recovers the table in the following order:

1. If the table does not exist in the database instance, it is created from the catalog information in the source table.

If the table already exists in the database instance, the catalog information in the source table and the target table is compared and adjusted (including integrity conditions and defaults).

2. The data is loaded (including LONGs).
3. Indexes are created if they exist in the source table.



The table is write-protected during the recovery. To remove the write-protection, back up the database instance after you have recovered the table.

## Result

The target table has been recovered completely with the data from the source file.

### Errors

If errors occur while a TABLELOAD command is being executed, the Replication Manager terminates the command and displays an error message.

## TABLEEXTRACT Command

### Syntax

`<tableextract_statement> ::= TABLEEXTRACT <part\_spec> \[Page 80\] <outfile\_spec> \[Page 80\]`



```
TABLEEXTRACT TABLE customer OUTFILE customer.data'
```

### Use

Use this [command for unloading data \[Page 31\]](#) to back up complete database tables ([source tables \[Page 15\]](#)), that is schema and data, in external files ([target files \[Page 15\]](#)). The format of the target files is specified by the internal database format. You cannot change the target files.

You can use the [TABLELOAD command \[Page 38\]](#) to recover the tables backed up to the file to the database.

### Prerequisites

When you use **TABLEEXTRACT TABLE** (see [part\\_spec \[Page 80\]](#)), the source table must exist in the database instance.

### Process Flow

The Replication Manager backs up

- The table catalog information (including index information, integrity conditions, and defaults)
- Data (including LONGS) in the internal database format

The table is write-protected during the backup.

When it processes the commands **TABLEEXTRACT ALL** and **TABLEEXTRACT USER** (see [part\\_spec \[Page 80\]](#)), the Replication Manager creates one or more target data files (depending on the size of the tables it is unloading), each of which can contain the data from one or more tables. The maximum size of a single target data file specified by the Replication Manager is 1 GB.

The files are created with the name of the target data file specified in the command. An extension of the form 0001 is added to the name. The maximum number of target data files for each **TABLEEXTRACT** is set at 9999.

As well as the target data files, the Replication Manager also creates another file with the extension 0000. This file logs the table names of the successfully unloaded tables and the names of the target data files. You cannot change this file after successfully completing the **TABLEEXTRACT**, since it specifies the order in which the tables are recovered and the assignment of table data to data files.

### Result

- Command **TABLEEXTRACT TABLE**: The source table is backed up completely to the target file.
- Commands **TABLEEXTRACT ALL/USER**: The source tables are backed up completely to the target files.

### Errors

If errors occur while a **TABLEEXTRACT** command is being executed, the Replication Manager terminates the command and displays an error message.



## Syntax Rules for Commands

The syntax used for [REPM Server commands \[Page 26\]](#) is the Backus Naur Form (BNF) with the following conventions:

```
<Rule name> ::= <Rule section>
```

The terms in angle brackets represent syntactical units. Each rule name must be resolved by a rule section.

A rule section is any combination of rules and endsymbols. Endsymbols are numbers, literals, or [keywords \[Page 84\]](#) and are not resolved further.

Rule sections set in angle brackets indicate an optional user input.

The keywords used in the commands are written in uppercase for the sake of clarity. They can be specified by the user in upper- or lowercase letters.

Blank characters are allowed in commands. These are not interpreted by the Replication Manager.

Comments in the [command file \[Page 13\]](#) are initiated by a slash at the start of the line. The entire line is then ignored when the command file is processed by the [REPM Server \[Page 13\]](#).

Endsymbols that start with **val** stand for values that must be specified by the user. Remember that values for '**valLITERAL**' (as specified in the syntax rules) must be placed in single quotation marks.

If you use keywords as names of columns, tables or users, you must place them in double quotation marks.

[Syntax Rules for Setting up a Session \[Page 41\]](#)

[Syntax Rules for Table Descriptions \[Page 42\]](#)

[Syntax Rules for Column Descriptions \[Page 50\]](#)

[Syntax Rules for Describing Data Access \[Page 68\]](#)

## Syntax Rules for Setting up a Session

The [rules \[Page 41\]](#) listed here are components of the [commands for setting up a session \[Page 27\]](#). These rules are explained in more detail.

You can go from the commands for setting up a session directly to their descriptions.

### database\_name\_statement

#### Syntax

```
<database_name_statement> ::= SERVERDB <valDBNAME> [ON <valDBNODE>]
```

<b>valDBNAME</b>	Name of the database instance, applies to the whole user session with this database instance.  If the name of the database instance is identical to a <a href="#">keyword [Page 84]</a> or if the upper- and lowercase notation is relevant, the name must be placed in quotation marks.
<b>valDBNODE</b>	Name of the host where the database instance you want to use is installed.

#### Use

This is a [syntax rule for setting up a session \[Page 41\]](#).

Use it to open a user session for the database instance specified with **valDBNAME**.

If you also specify the name of the host **valDBNODE**, on which this instance is installed, the user session for this instance is opened on this host. Otherwise, the Replication Manager assumes that the specified database instance is on the host where the user is logged on to the Replication Manager.

## sql\_mode

### Syntax

<sql\_mode> ::= INTERNAL | ANSI | DB2 | ORACLE

<b>INTERNAL</b>	Internal SQL dialect (system default) in the database system
<b>ANSI</b>	SQL dialect based on the ANSI standard (ANSI X3.135-1992, Entry SQL)
<b>DB2</b>	SQL dialect based on the definition for DB2 Version 4
<b>ORACLE</b>	SQL dialect based on the ORACLE7 definition

### Use

This is a [syntax rule for setting up a session \[Page 41\]](#).

The active SQL mode determines which SQL dialect is understood by the database system. It is transferred between the [REPM Server \[Page 13\]](#) and database instance when the connection is set up. The SQL statements and the [FASTLOAD command \[Page 32\]](#) in the [command file \[Page 13\]](#) are interpreted in accordance with the selected SQL mode. The SQL mode can change within a command file.

If you do not specify a mode, the internal SQL dialect in the database system will be used.

If you enter an invalid SQL mode, the database system automatically assumes that you want to use the internal SQL dialect.

## user\_statement

### Syntax

<user\_statement> ::= USER <valUSERNAME> <valPASSWORD>  
[<database\_name\_statement> [\[Page 41\]](#)]

valUSERNAME	User name
valPASSWORD	User password

The database system automatically converts the user name and password to uppercase letters. To suppress the conversion, you must place the names in quotation marks.

### Use

This is a [syntax rule for setting up a session \[Page 41\]](#).

The user data transferred is used to set up a user session with the database kernel.

## Syntax Rules for Table Descriptions

The [rules \[Page 41\]](#) listed here are components of the table description in the [commands for loading and unloading data \[Page 31\]](#). These rules are explained in more detail.

You can go from the description of the commands for loading and unloading data directly to the description of the rules used in each case.



If you want to use [keywords \[Page 84\]](#) as table or user names, you must place them in double quotation marks.

## **compare\_operator**

### Syntax

`<compare_operator> ::= < | > | = | <= | >= | !=`

### Use

This is a [syntax rule for describing tables \[Page 42\]](#).

You use [<simple\\_condition> \[Page 47\]](#) to specify the value with which a value in the data record is compared.

## **condition**

### Syntax

`<condition> ::= <simple\_condition \[Page 47\]> | (<condition>)  
| <condition> AND <condition> | <condition> OR <condition>  
| NOT <condition>`

### Use

This is a [syntax rule for describing tables \[Page 42\]](#).

The Replication Manager distinguishes between simple conditions and compound conditions. Simple conditions can be negated with **NOT**, combined with **AND** and **OR** to form compound conditions, or encapsulated as required.

Operators in parentheses are evaluated before those that are not in parentheses.

If no operators are in parentheses, the Replication Manager weights them as follows:

- **NOT** takes precedence over **AND** and **OR**
- **AND** takes precedence over **OR**
- If the weighting is identical, the operators are evaluated from left to right.

Only those records to which the simple or compound condition applies are loaded.



You want to load those data records from the source file `hotel.data` where the place is `BERLIN` and the price is less than `400.00` to the destination table `hotel`:

```
DATALOAD TABLE hotel
  IF POS 41-44 REAL < '400,00'
  AND
    POS 27-36 = 'BERLIN'
    hno      01-04 INTEGER
    name     09-18
    zip      20-25 DECIMAL
```

```

        place    27-36
        price    41-44 REAL
INFILE 'hotel.data' FORMATTED

```

## duplicates\_clause

### Syntax


<duplicates\_clause> ::= IGNORE DUPLICATES | REJECT DUPLICATES | UPDATE DUPLICATES

<b>IGNORE DUPLICATES</b>	The new line is not inserted.
<b>REJECT DUPLICATES</b>	The new line is rejected with an error message.
<b>UPDATE DUPLICATES</b>	The new line overwrites the existing line.

### Use

Use this [syntax rule for table descriptions \[Page 42\]](#) in a [DATALOAD command \[Page 34\]](#) to specify how to proceed when loading data from a [source file \[Page 14\]](#), and a line with the same key as the new line already exists in the [target file \[Page 15\]](#).

If you do not specify anything, **REJECT DUPLICATES** is the default.



```

DATALOAD TABLE customer
    UPDATE DUPLICATES
    cno      01-04
    surname  06-12
    zip      14-18
    city     20-31
INFILE 'customer.data' FORMATTED

```

## field\_format

### Syntax

<field\_format> ::= /\* blank \*/ | CHAR  
 | DECIMAL [<valFRACTION>] | ZONED [<valFRACTION>]  
 | INTEGER | REAL

<b>valFRACTION</b>	Number of characters after the comma for the data types DECIMAL and ZONED
--------------------	---

### Use

This is a [syntax rule for describing tables \[Page 42\]](#).

Use this option to describe the [data type \[Page 15\]](#) of the data fields in the [source file \[Page 14\]](#) or [target file \[Page 15\]](#).

You only need to specify the data type in a [command for loading or unloading data \[Page 31\]](#) or in a [condition \[Page 43\]](#) if the data field in question is to be read or output in a format other than CHAR. Every internal column format in the database can be read and output in CHAR format.

### Data Type Conversion During Loading or Unloading

	<a href="#">External data type [Page 16]</a>	<a href="#">Internal data type [Page 16]</a>
--	--	--

Loading data to a database instance	INTEGER, DECIMAL, ZONED, REAL	FIXED, FLOAT, SMALLINT, INTEGER
Unloading data from a database instance	FIXED, SMALLINT and INTEGER	INTEGER, DECIMAL, ZONED, REAL
	FLOAT	REAL or CHAR



```

DATALOAD TABLE article
  foa          01-08
  des          09-39
  stock        40-43 INTEGER
  min_ord      44-45 INTEGER
  ordered       46-49 INTEGER
  del_date     50-57
  price        58-65 DECIMAL (2)
  weight       66-69 REAL
INFILE 'customer.data' FORMATTED

```

## field\_pos

### Syntax

<field\_pos> ::= <valSTART\_POS> | <valSTART\_POS> - <valEND\_POS>

valSTART_POS	Start position of a data field
valEND_POS	End position of a data field

### Use

This is a [syntax rule for describing tables \[Page 42\]](#).

Use it to describe the following:

- The position of an input field in the [source file \[Page 14\]](#) when data is loaded
- The position of an output field in the [target file \[Page 15\]](#) when data is unloaded
- The position of a comparison value for loading data selectively ([simple condition \[Page 47\]](#)).

The position depends on the [format \[Page 18\]](#) of the data that you want to load or unload.

### Format FORMATTED

If the data file has the format [FORMATTED \[Page 19\]](#), only enter absolute start and end positions for the data fields in the data file.



The data file `customer.data` has the FORMATTED format. The data fields have a standard format and end with a line break.

Positio n no.	1 2 3 4 5 6 7 8 9 .....
	0 1 m o o r e - 1 0 0 4 4 N e w   Y o r k - - - - -
	0 2 s c h m i d t - 1 3 4 0 3 B e r l i n - - - - -
	0 3 s o r o k i n - 9 0 1 8 5 L o s   A n g e l e s

FASTLOAD command:

```
FASTLOAD TABLE customer
  cno      01-02
  surname  03-10
  zip      11-15
  place    16-27
INFILE 'customer.data' FORMATTED
```

## Format COMPRESSED

If the data file has the format [COMPRESSED \[Page 18\]](#), only enter relative positions for the data fields in the data file.



The data file `customer.data` has the COMPRESSED format. The data fields do not have a standard format but are separated by commas:

Position no.	1	2	3	4	....
	001,moore,10044,New York				
	002,schmidt,13403,Berlin				
	003,sorokin,90185,Los Angeles				

FASTLOAD command:

```
FASTLOAD TABLE customer
  cno      1
  surname  2
  zip      3
  place    4
INFILE 'customer.data'
```

COMPRESSED is the default format used by the Replication Manager.



The data file `customer.data` has the FORMATTED BINARY format.

Position no.	1	2	3	4	5	6	7	8	9	.....
	0 1 m o o r e x 1 0 0 4 4 N e w Y o r k : : : 0 2 s c h m i d t : 1 3 4 0 3 B e r l i n : : : : : 0 3 s o r o k i n 9 0 1 8 5 L o s   A n g e l e s									

The data fields have a standard format but do not end with a line break.

FASTLOAD command:

```
FASTLOAD TABLE customer
  cno      01-02
  surname  03-10
  zip      11-15
  place    16-26
INFILE 'customer.data' FORMATTED BINARY
```



## if\_condition

### Syntax

```
<if_condition> ::= IF <condition [Page 43]> | OTHERWISE
```

## Use

This is a [syntax rule for describing tables \[Page 42\]](#).

Use it to define the selection criterion that determines which records from the source file are loaded to which target table.

You can selectively load data records that do not fulfill the selection criterion to a specific target table by using the **OTHERWISE** condition. This means that you can load all those records in the source table, for which you cannot define a standard selection criterion, to a certain target table.

This syntax rule allows you to carry out the following actions when [data is loaded \[Page 31\]](#) from a [source file \[Page 14\]](#) to a [target table \[Page 15\]](#):

- [Loading Data to Multiple Tables Simultaneously \[Page 10\]](#)
- [Selecting Data Records from the Source File \[Page 12\]](#)



Only specify OTHERWISE for the **last** table in a series of [DATALOAD commands \[Page 34\]](#).

## restore\_spec

### Syntax

**<restore\_spec> ::= FOR DATALOAD | FOR FASTLOAD | FOR DATAUPDATE**

### Use

Use this [syntax rule for describing tables \[Page 42\]](#) in a [DATAEXTRACT command \[Page 36\]](#) to specify which command for loading data in the [command file \[Page 13\]](#) to generate.

If you want to generate a [FASTLOAD command \[Page 32\]](#), this command has the value 80% as the usage of the database (**<usage\_spec>** [\[Page 49\]](#)). This is the default used by the Replication Manager.



```
DATAEXTRACT FOR DATALOAD TABLE article
OUTFILE 'article.control'
OUTFILE 'article.data'
```

## simple\_condition

### Syntax

**<simple\_condition> ::= POS [<field\\_pos> \[Page 45\]](#) [<field\\_format> \[Page 44\]](#) [HEX] [<compare\\_operator> \[Page 43\]](#) '<valLITERAL>'**

POS	<a href="#">Key word [Page 84]</a>
field_pos	Position of a value in the data record
field_format	<a href="#">Data type [Page 15]</a> of the value
compare_operator	Compare operator
valLITERAL	Constant

As with the other fields in a data record, you use the position of a value that you want to compare to describe it. You only need to specify the value format if it is not CHAR.

You specify a constant as a [plain text value \[Page 17\]](#) and place it in quotation marks. The constant is converted to the data type of the value that you want to compare in the data record.

If the constant you want to use as a comparison value is a number, it must have a valid number format, that is, it must be a floating point number in mantissa/exponent notation, a fixed-point number with the currently defined decimal representation (or default decimal representation in the Replication Manager).

## Use

This is a [syntax rule for describing tables \[Page 42\]](#).

Use it to define the selection criterion that determines which records from the [source file \[Page 14\]](#) are loaded to which [target table \[Page 15\]](#). The data records that are you want to load are selected by comparing them with a constant.



You want to load only those data records from the source file `hotel.data` where the price is less than `400.00` to the destination table `hotel`:

```
DATALOAD TABLE hotel
  IF POS 41-44 REAL < '400.00'
    hno      01-04 INTEGER
    name     09-18
    zip      20-25 DECIMAL
    place    27-36
    price    41-44 REAL
INFILE 'hotel.data' FORMATTED
```

## table\_name

### Syntax

**<table\_name> ::= <valTABLE\_NAME> | <valTABLE\_OWNER>.<valTABLE\_NAME>**

<b>valTABLE_NAME</b>	Name of the table Specify the table name in accordance with the SQL conventions. You can also define a user name as a prefix.
<b>valTABLE_OWNER</b>	Owner of the table

## Use

This is a [syntax rule for describing tables \[Page 42\]](#).

Use it in a [command for loading or unloading data \[Page 31\]](#) to specify a table name.



```
FASTLOAD TABLE hotel
  IF (POS 41-44 REAL < '400.00')
  AND
    (POS 41-44 REAL >= '400.00')
    hno      01-04 INTEGER
    name     09-18
    zip      20-25 DECIMAL
    place    27-36
    price    41-44 REAL
INFILE 'hotel.data' FORMATTED;
```



```

DATALOAD TABLE berolina.hotel
  IF (POS 41-44 REAL < '400.00')
  AND
    (POS 41-44 REAL >= '400.00')
    hno      01-04 INTEGER
    name     09-18
    zip      20-25 DECIMAL
    place    27-36
    price    41-44 REAL
INFILE 'hotel.data' FORMATTED;

```

## table\_spec

### Syntax

**<table\_spec>** ::= [<table\\_name> \[Page 48\]](#) [<if\\_condition> \[Page 46\]](#)

### Use

This is a [syntax rule for describing tables \[Page 42\]](#).

Use this rule in a [DATALOAD command \[Page 34\]](#) to specify the name(s) and contents of the [target table\(s\) \[Page 15\]](#).

It also applies to loading a table in the [FASTLOAD command \[Page 32\]](#). However, data can only be loaded to **one** table with **one** FASTLOAD command.



You want to load those data records in the `address.data` source file that begin with the letter '**k**' to the target table with the name `customer` and all those data records that begin with the letter '**p**' to the target table with the name `partner`:

```

DATALOAD TABLE customer      IF POS 1 = 'k'
  cno      2
  surname   3
  zip       5
  place     6
DATALOAD TABLE partner      IF POS 1 = 'p'
  cno      2
  surname   3
  zip       5
  place     6
INFILE 'address.data'

```

## usage\_spec

### Syntax

**<usage\_spec>** ::= WITH valUSAGE % USAGE | WITH valUSAGE ROWS USAGE

### Use

Use this [syntax rule for describing tables \[Page 42\]](#) in a [FASTLOAD command \[Page 32\]](#) to specify the extent to which you want to fill a table page with data records.

To do this, you can specify a percentage between 50 and 100 or define the number of rows (data records) that you want to load in each table page. If you specify a number of rows that exceeds the

actual number possible, the Replication Manager displays an error message during the load operation, indicating the maximum number of rows permitted in each table page.

The default setting in the Replication Manager is 80 %.

If the table is not modified at all, or only slightly, it is a good idea to utilize the occupied memory by more than 80%.

If considerable dynamic growth is anticipated for the table, it is a good idea to utilize the occupied memory by less than 80%.



The specified memory utilization does not guarantee that it will actually be reached. Use the Database Manager program to check the current memory utilization for the relevant table.



You want to specify a memory utilization of 100% to FASTLOAD data from the source file customer.data to the customer table.

**FASTLOAD with 100 % USAGE**

```
TABLE customer
  cno          1
  surname      2
  zip          3
  place        4
INFILE 'customer.data'
```



You want to enter 25 data records in each table page on the database instance.

**FASTLOAD with 25 ROWS USAGE**

```
TABLE customer
  cno          1
  surname      2
  zip          3
  place        4
INFILE 'customer.data'
```



## Syntax Rules for Column Descriptions

The [rules \[Page 41\]](#) listed here are components of the column description in the [commands for loading and unloading data \[Page 31\]](#). These rules are explained in more detail.

You can go from the description of the commands for loading and unloading data directly to the description of the rules used in each case.



If you want to use [keywords \[Page 84\]](#) as column names, you must place them in double quotation marks.



## acc\_column\_spec

### Syntax

**<acc\_column\_spec> ::= <[key column spec \[Page 56\]](#)> | <[simple column spec \[Page 67\]](#)>**

## Use

This is a [syntax rule for describing columns \[Page 50\]](#).

You use it in a [DATAUPDATE command \[Page 37\]](#) to define whether the [qualification columns \[Page 15\]](#) are key columns or non-key columns of the target table.

If you use key columns as qualification columns ([key\\_column\\_spec \[Page 56\]](#)), each data record in the source file updates that line in the target table that matches the specified key.



```
DATAUPDATE TABLE customer
KEY   cno 01-04
SET    place 05-16
INFILE 'customer.data' FORMATTED
```

If you use non-key columns as qualification columns ([simple\\_column\\_spec \[Page 67\]](#)), each data record in the source file updates each line in the target table that is identified with the value(s) of the qualification column(s).



```
DATAUPDATE TABLE customer
      place 1
SET    zip 2
INFILE 'customer.data'
```

## acc\_column\_spec\_mlt

### Syntax

<acc\_column\_spec\_mlt> ::= <[acc\\_column\\_spec \[Page 50\]](#)> <acc\_column\_spec\_mlt>

## Use

This is a [syntax rule for describing columns \[Page 50\]](#).

You use it in a [DATAUPDATE command \[Page 37\]](#) to define the [qualification columns \[Page 15\]](#).

You can use both key columns and non-key columns as qualification columns.

The qualification columns must be listed before the target columns ([set\\_column\\_spec \[Page 66\]](#)) in a DATAUPDATE command.

## column\_assignment

### Syntax

<column\_assignment> ::= <valCOLUMN\_NAME> <'valLITERAL'>  
| <valCOLUMN\_NAME> <[generate\\_spec \[Page 56\]](#)>

### Loading Constants: <valCOLUMN\_NAME> <'valLITERAL'>

To load any constant, specify the constant in a position. Place the value in single quotation marks.

The Replication Manager handles the constant as an input value in plain text, and converts it into the format of the target column.

If you want to load the constant into a numeric column, it must have a valid numeric format ([plain text value \[Page 17\]](#), [binary value \[Page 17\]](#)).



```

DATALOAD TABLE article
  foa          01-08 CHAR
  des          09-39 CHAR
  stock        40-43 INTEGER
  min_order    50-45 INTEGER
  ordered      46-49 INTEGER
  del_date     50-57 CHAR
  price        58-65 DECIMAL (2)
  weight       66-69 REAL
INFILE 'customer.data' FORMATTED

```

## Loading Special Constants: <valCOLUMN\_NAME> <generate\_spec >

As well as loading free constants, you can also load the special values **STAMP**, **USER**, **USERGROUP**, **DATE**, **TIME**, **TIMESTAMP**, **TRUE** and **FALSE**.



```

DATALOAD TABLE article
  changed_by USER
  changed_on DATE
  changed_at TIME
  foa          1
  des          2
  stock        3 NULL IF POS 3 < '0'
INFILE 'customer.data'

```

## Use

This is a [syntax rule for describing columns \[Page 50\]](#).

When you [load data \[Page 31\]](#), you can use this rule to specify that a constant value is loaded into the specified column for each loaded data record, and which value this constant must have.

To do this, specify the constant in a position in the command. No data will then be loaded from the data file for the appropriate column.



If the [source file \[Page 14\]](#) is empty, the constants specified in the command will not be loaded.



## column\_descriptor

### Syntax

<column\_descriptor> ::= <valCOLUMN\_NAME> [<field\\_pos> \[Page 45\]](#) [<format\\_spec> \[Page 55\]](#)

<b>valCOLUMN_NAME</b>	<p>Column name Specify it according to the SQL conventions. It can also contain the table name as a prefix (for example, CUSTOMER.CNO).</p> <p>The columns can be assigned special values (date, time, timestamp, Boolean values) and NULL or default values can be loaded.</p>
-----------------------	---

## Use

This is a [syntax rule for describing columns \[Page 50\]](#).

You use it in a [command for loading data \[Page 31\]](#) to assign a data field in the [source file \[Page 14\]](#) to a column in the [target table \[Page 15\]](#). To do this, you specify the column name, the position, and the data type of the data in the source file.



```
DATALOAD TABLE article
  foa          01-08 CHAR
  des          09-39 CHAR
  stock        40-43 INTEGER
  min_ord      44-45 INTEGER
  ordered      46-49 INTEGER
  del_date     50-57 CHAR
  price        58-65 DECIMAL (2)
  weight       66-69 REAL
INFILE 'article.data' FORMATTED
```

## column\_id

### Syntax

<column\_id> ::= <valCOLUMN\_NAME> | <valCOLUMN\_ID>

<b>valCOLUMN_NAME</b>	Column name Specify it according to the SQL conventions. It can also contain the table name as a prefix (for example, CUSTOMER.CNO).
<b>valCOLUMN_ID</b>	Column ID

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

You use it in a [command for unloading data \[Page 31\]](#) to assign a column in the [source file \[Page 15\]](#) to a data field in the [target table \[Page 15\]](#).



```
DATAEXTRACT cno, last name, zip, place FROM customer
OUTFIELDS
  cno          01-04
  last name    06-12
  zip          14-18
  place        20-31
OUTFILE 'customer.data' FORMATTED
```

The column number **valCOLUMN\_ID** indicates the position of the column in the SELECT statement.



```
DATAEXTRACT cno, last name, zip, place FROM customer
OUTFIELDS
  1  01-04
  2  06-12
  3  14-18
  4  20-31
OUTFILE 'customer.data' FORMATTED
```

## column\_id\_spec

### Syntax

`<column_id_spec> ::= <column_id> [Page 53] <field_pos> [Page 45] <format_spec> [Page 55] <null_assign> [Page 59]`

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use this rule in a [command for unloading data \[Page 31\]](#) to assign the data fields of the [target file \[Page 15\]](#) to the columns of the [source table \[Page 15\]](#). You specify the [external data types \[Page 16\]](#) and the conditions for unloading the data.

### Note the following rules:

Decide whether you want your column descriptions to contain `field_pos` field descriptions.

However, you must specify position descriptions for **all or none** of the described columns. If you do not, the Replication Manager generates an error and terminates the command.

### Output Format COMPRESSED

- Enter the position descriptions as relative positions only.

If you enter position descriptions with start and end positions for single columns or all columns, the Replication Manager generates an error and terminates the command.



```
DATAEXTRACT * from customer
OUTFIELDS
  cno          1
  last name    2
  zip          3
  place        4-31
OUTFILE 'customer.data' COMPRESSED
```

- Assign position 1 to the first column in your list. The position numbers of the following columns rise by 1 each time. This also means that each position number can only be assigned once. If you do not keep to these rules, the Replication Manager generates an error and terminates the command.



```
DATAEXTRACT * from customer
OUTFIELDS
  cno          1
  last name    2
  street       3
  zip          3      ERROR
  place        5      ERROR
OUTFILE 'customer.data' COMPRESSED
```

- You can assign a column to multiple **different** positions.



```
DATAEXTRACT * from customer
OUTFIELDS
  cno          1
  last name    2
  zip          3
  place        4
```

```

      last name 5
OUTFILE 'customer.data' COMPRESSED

```

### Output Format FORMATTED

- Specify all position descriptions with exact start and end positions.
- Specify only ascending, non-overlapping values for position descriptions.  
If the positions do not follow on from each other, the REPM Server fills the gaps with space characters. This also applies to binary data.



```

DATAEXTRACT * from customer
OUTFIELDS
  cno          01-04
  last name    06-12
  zip          14-18
  place       16-31
OUTFILE 'customer.data' FORMATTED

```

- Define the positions for the target file at least as long as the length of the values in the database.  
If you define a position for the target file that is **longer** than the length of the value in the database, the following occurs:
  - Character strings are aligned left and space characters entered to make them the correct length
  - Numeric values are aligned right and space characters entered to make them the correct length
 If you define a position for the target file that is **shorter** than the length of the values in the database, the Replication Manager generates an error and terminates the command.

## column\_names

### Syntax

```
<column_names> ::= <valCOLUMN_NAME> | <valCOLUMN_NAME>,<column_names>
```

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use this rule in a [DATEXTRACT command \[Page 36\]](#) with `order_clause` (see variant 2 of the DATEXTRACT command) to specify the sort order of the columns that you want to unload.



```

DATAEXTRACT FOR DATALOAD TABLE article ORDER BY delivery date,
price
OUTFILE 'article.command'
OUTFILE 'article.data'

```

## format\_spec

### Syntax

```
<format_spec> ::= <field_format [Page 44]> [HEX] [<numerical_functions [Page 61]>]
```

## Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [command for loading or unloading data \[Page 31\]](#) to specify the data fields in the [source file \[Page 14\]](#) or [target file \[Page 15\]](#) (data type, position, and so on).

## generate\_spec

### Syntax

```
<generate_spec> ::= USER | USERGROUP
| STAMP | DATE | TIME | TIMESTAMP
| <sequence_number> [Page 67]
```

<b>USER</b>	Name of the current user is loaded
<b>USERGROUP</b>	Name of the group of the current user is loaded.  If this user is not assigned to a user group, the name of the current user is loaded.  The column into which you want to load one of these values must have the type CHAR (n) where n > 32.
<b>DATE</b>	Current date  The column into which you want to load one of these values must have the type DATE.
<b>TIME</b>	Current time  The column into which you want to load one of these values must have the type TIME.
<b>TIMESTAMP</b>	Current timestamp  The column into which you want to load one of these values must have the type TIMESTAMP.
<b>STAMP</b>	A unique value generated by SAP DB, which can only be loaded into columns of the type CHAR (n) BYTE where n > 8.

## Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it to load the specified special constants into the database.

## key\_column\_spec

### Syntax

```
<key_column_spec> ::= KEY <simple_column_spec [Page 67]>
```

## Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it to specify that a certain [qualification column \[Page 15\]](#) in the [DATAUPDATE command \[Page 37\]](#) is a key column of the target table.



## lit\_column\_spec

### Syntax

`<lit_column_spec> ::= '<valLITERAL>' <field_pos [Page 45]>`

'<valLITERAL>'	Text constant
----------------	---------------

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it to define text constants for output in addition to the output values in the [DATAEXTRACT command \[Page 36\]](#).



```
DATAEXTRACT * from customer
OUTFIELDS
  'customer number:' 01-15
  cno                16-19
  'Name:'            20-25
  last name          26-32
  zip                34-38
  place              50-61
OUTFILE 'customer.data' FORMATTED
```

The text constant is specified in single quotation marks instead of the column name or column ID ([column\\_id \[Page 53\]](#)).

## load\_column\_spec

### Syntax

`<load_column_spec> ::= <column_descriptor> [Page 52] [ <null_condition> [Page 60] ]  
| <column_assignment> [Page 51]`

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [command for loading data \[Page 31\]](#) to assign the data fields in the [source file \[Page 14\]](#) to the columns in the [target table \[Page 15\]](#), and specify the [external data types \[Page 16\]](#) and conditions for loading the data.



```
DATALOAD TABLE article
  foa      01-08 CHAR
  des      09-39 CHAR
  stock    40-43 INTEGER
  min_ord  44-45 INTEGER
  ordered  46-49 INTEGER
  del_date 50-57 CHAR
  price    58-65 DECIMAL (2)
  weight   66-69 REAL
INFILE 'article.data' FORMATTED
```

## load\_column\_spec\_mlt

### Syntax

```
<load_column_spec_mlt> ::= /* leer */
| <load_column_spec> [Page 57] <load_column_spec_mlt>
```

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [command for loading data \[Page 31\]](#) to describe the data records you want to load from a [source file \[Page 14\]](#) and their assignment to the columns of the [target table \[Page 15\]](#).



Table definition:

```
create table customer (cno char (4), surname char (6) NOT NULL,
zip integer, place char (11), PRIMARY KEY (cno))
```

Load command:

```
DATALOAD TABLE customer
  cno      01-04
  last name 06-12
  zip      14-18
  place    20-31
INFILE 'customer.data' FORMATTED
```

### Rules

Data must exist in the source file for each column that you specify in the data load command.

If you do not specify columns in the target table in the command, the entire column is populated with the default value defined for this column during the load operation. The NULL value is loaded if no specific default value is defined for the column.



```
DATALOAD TABLE customer
  cno      01-04
  last name 06-12
INFILE 'customer.data' FORMATTED
```

zip and place are populated with the default value when the data is loaded

Key columns and mandatory columns (columns that are defined as NOT NULL without a default value) must be specified in the data load command. Otherwise, the command terminates with an SQL error.



```
DATALOAD TABLE customer
  last name 06-12
  zip      14-18
  place    20-31
INFILE 'customer.data' FORMATTED
```

The command terminates with an SQL error.

If you do not specify any columns for the target table in the data load command, the table is loaded as if all columns in the target table were specified in the command. If this is the case, data must exist in the source file for all of the columns in the target table.



```
DATALOAD TABLE customer
INFILE 'customer.data' FORMATTED
```

Data must exist in the source file for all of the columns in the target table.

## null\_assign

### Syntax

```
<null_assign> ::= [IF] NULL SET '<valLITERAL>'
```

'<valLITERAL>'	Null value representation
----------------	---------------------------

Place the value for the null representation in single quotation marks. This is a [plain text value \[Page 17\]](#) that the Replication Manager represents as a character string or binary numeric value. This depends on the [external data type \[Page 16\]](#) of the column for which you want to generate the null value.

If you want to display the null value representation in one of the [binary \[Page 17\]](#) external data formats, it must have a valid numeral format. This means either a floating decimal number in mantissa/exponent representation, with the agreed or standard decimal setting.

The generated null value representation is written to the same place in the file as the actual column value.

If the null value representation is shorter than the length of the value specified by the position, the value is filled with space characters. If the null value representation is longer, it is shortened to the specified length. The Replication Manager generates a warning.

If columns in the [target table \[Page 15\]](#) permit NULL values and you have not specified a null value representation, the Replication Manager default for unloaded NULL values is used ([null\\_spec \[Page 80\]](#)). This has the external data type CHAR.

If you define a null value representation for unloading NOT-NULL columns, it does not cause an error.

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [command for unloading data \[Page 31\]](#) to specify which value for the data field of the [source table \[Page 15\]](#) is written to the [target file \[Page 15\]](#) if the value in the source table is a NULL value.



You want to unload the source table `article` from the database into the target file `article.data`. Some of the columns in the table contain NULL values.

Define a separate condition for each of these columns. If this condition is met, the corresponding null value representation is entered in the target file.

```
DATAEXTRACT * FROM article
OUTFIELDS
  foa          01-08
  des          09-39      IF NULL SET ' '
  stock       40-43 INTEGER IF NULL SET '0'
  min_ord     44-45 INTEGER
  price       46-53 DECIMAL (2) IF NULL SET 'X'
  weight      54-57 REAL
OUTFILE 'article.data' FORMATTED
```



You want to unload the source table `article` from the database into the target file `article.data`. Some of the columns in the table contain NULL values. Because there is no null value representation defined for the column in the command, but all values have the external data type CHAR, the Replication Manager default null value representation `'?'` is used.

```
DATAEXTRACT * FROM article
OUTFIELDS
  foa      01-08
  des      09-39
  stock    40-51
  min_ord  52-63
  price    64-74
  weight   75-85
OUTFILE 'article.data' FORMATTED
```



You want to unload the source table `article` from the database into the target file `article.data`. Some of the columns in the table contain NULL values. Because there is no null value representation defined for the column in the command, the Replication Manager attempts to use the null value representation defined as a default (data type CHAR). This causes an error if a numeric external data type has been defined for the columns. The command terminates with an error message.

```
DATAEXTRACT * FROM article
OUTFIELDS
  foa      01-08
  des      09-39
  stock    40-43 INTEGER
  min_ord  44-45 INTEGER
  price    46-53 DECIMAL (2)
  weight   54-57 REAL
OUTFILE 'article.data' FORMATTED
```

## null\_condition

### Syntax

```
<null_condition> ::= NULL [IF] <condition [Page 43]> | DEFAULT NULL
```

When the condition is evaluated, the shorter of the comparison values (value in the [source file \[Page 14\]](#) or the null) is filled with space characters.

A check is made before each line in the source file is loaded to see whether the condition formulated for the columns applies. If it does, the NULL value is inserted in this column. If not, the value from the assigned field in the source file is inserted.

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [command for loading data \[Page 31\]](#) to specify under which conditions the NULL value is to be loaded into a column of the [target table \[Page 15\]](#).

You can use the **DEFAULT-NULL** condition if the NULL value in the source file is represented in the same way for all columns that you want to load from the target table. The character string that you need to specify after **NULL IF POS** is only specified once as file option [NULL \[Page 80\]](#) in the **DEFAULT-NULL** condition.



You cannot load columns defined as key columns (KEY) or NOT NULL with the NULL value. If you do, the table load action terminates with an appropriate error message.

If, when you created the table in the database, you defined a default other than NULL for columns, you cannot load NULL values into the columns. If this is the case, the Replication Manager uses the default defined for the column instead of the NULL value.



You want to load the data from the source file `article.data` into the database. You want to load the NULL value into some columns of the target table. Define a separate condition for each of these columns. The NULL value is entered if this condition is met.

```
DATALOAD TABLE article
  foa          01-08
  des          09-39          NULL IF POS 09-11 = ' '
  stock        40-43 INTEGER NULL IF POS 40-43 INTEGER < '0'
  min_ord      44-45 INTEGER
  price        46-53 DECIMAL (2) NULL IF POS 1 <> 'X'
                                   OR POS 46-53 DECIMAL < '0'
  weight       54-57 REAL
INFILE 'article.data' FORMATTED
```



You want to load the data from the source file `article.data` into the database. You want to load the NULL value into some columns of the target table. The same '?' representation of the NULL value applies to all columns in the command where **DEFAULT NULL** is specified.

```
DATALOAD TABLE article
  foa          01-08
  des          09-39 DEFAULT NULL
  stock        40-43 INTEGER DEFAULT NULL
  min_ord      44-45 INTEGER
  price        46-53 DECIMAL (2) DEFAULT NULL
  weight       54-57 REAL
INFILE 'article.data' FORMATTED
NULL '?'
```

## numerical\_functions

### Syntax

```
<numerical_functions> ::= <scale_spec> [Page 64]
| <round_or_trunc_spec> [Page 64]
| <scale_spec> <round_or_trunc_spec>
```

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it to scale, round off or shorten numeric data values when you load data from the [source file \[Page 14\]](#) to the [target table \[Page 15\]](#), or unload data from the [source table \[Page 15\]](#) to the [target file \[Page 15\]](#).



Note that you must always specify `scale_spec` before `round_or_trunc_spec`.



```

DATALOAD TABLE distance
...
cm 7 SCALE 2
cm 7 SCALE -3 ROUND 1
...
INFILE 'meter.data'

```



```

DATAEXTRACT * FROM distance
cm      10-14 INTEGER SCALE 2
m       14-17 INTEGER
km      18-21 INTEGER SCALE -3 TRUNC 2

OUTFILE 'dimensions.bin' FORMATTED

```

## order\_clause

### Syntax

**<order\_clause>** ::= ORDER BY [<column\\_names \[Page 55\]>](#)

### Use

Use this [syntax rule for column description \[Page 50\]](#) as a supplement to the DATAEXTRACT FOR DATALOAD command (variant 2 of the [DATAEXTRACT command \[Page 36\]](#)) to specify the sort order of the unloaded data.



This is an element of the SQL language.

**See also:** *Reference Manual: SAP DB 7.2 and 7.3*

## output\_column

### Syntax

**<output\_column>** ::= [<column\\_id\\_spec> \[Page 54\]](#) | [<lit\\_column\\_spec> \[Page 57\]](#)

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [command for unloading data \[Page 31\]](#) to assign the columns in the [source table \[Page 15\]](#) to the data fields in the [target table \[Page 15\]](#), and specify the [external data types \[Page 16\]](#) and conditions for unloading the data.



```

DATALOAD TABLE article
foa      01-08 CHAR
des      09-39 CHAR
stock    40-43 INTEGER
min_ord  44-45 INTEGER
ordered  46-49 INTEGER
del_date 50-57 CHAR

```

```

price      58-65 DECIMAL (2)
weight     66-69 REAL
INFILE 'article.data' FORMATTED

```

## output\_column\_list

### Syntax

```

<output_column_list> ::= <output_column> [Page 62] | <output_column>
<output_column_list>

```

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [DATAEXTRACT command \[Page 36\]](#) to describe the unloaded columns of a [source table \[Page 15\]](#) and the representation of the data records in the [target file \[Page 15\]](#).

### Note the following rules:

- The columns can be in any order in the column description.
- The column list can only contain columns from the SELECT statement ([select expression \[Page 65\]](#)) or a subset of them.
- Values are output only for those columns in the column list. If the column list contains more columns than the SELECT statement, the Replication Manager generates an error and terminates the command.
- If you do not describe the columns of the source table in the command, you do not need to specify the OUTFIELDS key word. The data is output in plain text.
- If you have defined the format COMPRESSED for the target file, the values of the unloaded columns are output in the order in the SELECT statement, and divided with separators.
- If you have defined the format FORMATTED for the target file, the values of the unloaded columns are formatted and output in the order in the SELECT statement. The length of the individual output values depends on the defined sizes of the individual columns in the source table.
- If the SELECT statement does not contain a column name (**DATAEXTRACT \* FROM ...**), the columns are formatted according to the format of the target file. They are output in the order specified by the database for processing the command.



Definition of the source table:

```

create table customer (cno char (4), surname char (6) NOT NULL,
zip integer, place char (11), PRIMARY KEY (cno))

```

Unload command:

```

DATAEXTRACT * from customer
OUTFIELDS
  cno      01-04
  last name 06-12
  zip      14-18
  place    20-31
OUTFILE 'customer.data' FORMATTED

```

## **round\_or\_trunc\_spec**

### Syntax

**<round\_or\_trunc\_spec> ::= ROUND <valFRACTION> | TRUNC <valFRACTION>**

<b>valFRACTION</b>	Number of decimal places  The value must be between 0 and 18. This function does not have any effect if the number does not have any decimal places.
--------------------	--

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it to specify the number of decimal places in a number.

#### **ROUND <valFRACTION>**

The value is rounded off at the (<valFRACTION>+1)th decimal place. If this number is  $\geq 5$ , the value is rounded up. If it is  $< 5$ , the value is rounded down. The result is a number in which the (<valFRACTION>+1)th and all subsequent decimal places are equal to 0. The other digits in the number may have been changed if the value was rounded up.



```

DATALOAD TABLE distance
...
cm 7 SCALE 2
cm 7 SCALE -3 ROUND 1
...
INFILE 'meter.data'

```

#### **TRUNC <valFRACTION>**

The (<valFRACTION>+1)th and all subsequent decimal places of the value are set to 0. The first <valFRACTION> decimal places remain unchanged.



```

DATAEXTRACT * FROM distance
cm      10-14 INTEGER SCALE 2
m       14-17 INTEGER
km      18-21 INTEGER SCALE -3 TRUNC 2

OUTFILE 'dimensions.bin' FORMATTED

```

## **scale\_spec**

### Syntax

**<scale\_spec> ::= SCALE <valSCALE\_FACTOR>**

<b>valSCALE_FACTOR</b>	Scaling factor, can be positive or negative  The value to which the syntax rule refers is multiplied by the corresponding power of ten.
------------------------	---

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it to scale values.



You can combine this syntax rule with [round\\_or\\_trunc\\_spec \[Page 64\]](#).



```
DATALOAD TABLE distance
```

```
...
```

```
cm 7 SCALE 2
```

```
cm 7 SCALE -3
```

```
...
```

```
INFILE 'meter.data'
```



```
DATAEXTRACT * FROM
```

```
cm      10-14 INTEGER SCALE 2
```

```
m       14-17 INTEGER
```

```
km      18-21 INTEGER SCALE -3
```

```
OUTFILE 'dimensions.bin' FORMATTED
```

## **select\_expression**

### Syntax

<select\_expression>

This database query is formulated in the same way as a SELECT statement in SQL, only the keyword **DATAEXTRACT** replaces SELECT. All options of the SELECT statement are permitted.

- Select the result columns and determine their order in the result table
- Join multiple tables
- Use qualifications to select result lines
- Specify sort order
- Specify locks (WITH LOCK)

### Use

Use this [syntax rule for describing columns \[Page 50\]](#) in a [DATAEXTRACT command \[Page 36\]](#) to specify which columns in a table to unload.



```
DATAEXTRACT * FROM article
```

```
OUTFIELDS
```

```
foa      01-08
```

```
des      09-39
```

```
stock    40-51
```

```
min_ord  52-63
```

```
price    64-74
```

```
weight   75-85
```

```
OUTFILE 'article.data' FORMATTED
```



```
DATAEXTRACT ano,des,stock,min_ord FROM article order by ano
```

```
OUTFIELDS
```

```
foa      1
```

```
des      2
```

```
stock    3
```

```

    min_ord    4
OUTFILE 'article.data'

```



```

DATAEXTRACT ano,des,stock,min_ord FROM article WITH LOCK
OUTFIELDS
    foa        1
    des        2
    stock      3
    min_ord    4
OUTFILE 'article.data'

```

## set\_column\_spec

### Syntax

**<set\_column\_spec> ::= SET <[load\\_column\\_spec \[Page 57\]](#)>**

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [DATAUPDATE command \[Page 37\]](#) to define the columns of the [target table \[Page 15\]](#) that you want to update.



```

DATAUPDATE TABLE customer
KEY   cno 01-04
SET   place 05-16
INFILE 'customer.data' FORMATTED

```

## set\_column\_spec\_mlt

### Syntax

**<set\_column\_spec\_mlt> ::= <[set\\_column\\_spec \[Page 66\]](#)> <set\_column\_spec\_mlt>**

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [DATAUPDATE command \[Page 37\]](#) to define the columns of the [target table \[Page 15\]](#) that you want to update.



```

DATAUPDATE TABLE customer
KEY   cno 01-04
SET   place 05-16
INFILE 'customer.data' FORMATTED

```

## **sequence\_number**

### Syntax

`<sequence_number> ::= SEQNO | SEQNO <valSTART>  
| SEQNO <valSTART> <valINCREMENT>`

<b>valSTART</b>	Start value and first value to be loaded
<b>valINCREMENT</b>	Increment between loaded values

The keyword **SEQNO** specifies that sequential numbers are generated. Both the start value and the increment can be negative.

If you only enter the keyword **SEQNO** in the command, the start value is 0 and the increment between values is 1.

If you only enter the start value in the command after **SEQNO**, the increment between the values is 1.


### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it to load generated sequential numbers with a specified start value and increment.

The set of calculable numbers is cyclical. This means that the value that comes after the largest possible value is the smallest possible value. In this way, you can generate as many numbers as you want, including repeated sequential numbers.

A column of the type **FIXED(10)** is enough for storing the sequential numbers.



```

DATALOAD TABLE article
  foa          SEQNO 10000 5
  des          2
  stock        3 NULL IF POS 3 < '0'
INFILE 'article.data' FORMATTED

```

## **simple\_column\_spec**


### Syntax

`<simple_column_spec> ::= <column descriptor \[Page 52\]> | <column assignment \[Page 51\]>`

### Use

This is a [syntax rule for describing columns \[Page 50\]](#).

Use it in a [DATAUPDATE command \[Page 37\]](#) to assign the data fields of the [qualification column \[Page 15\]](#) in the [source file \[Page 14\]](#) to the columns of the [target table \[Page 15\]](#), and specify the [external data types \[Page 16\]](#).



```

DATAUPDATE TABLE article
  foa 01-08          CHAR
  des 09-39          CHAR
SET min_ord 40-41    INTEGER
SET delivery date 42-49 CHAR
INFILE 'article.data' FORMATTED

```

## Syntax Rules for Describing Data Access

The [rules \[Page 41\]](#) listed here are components of the [source file \[Page 14\]](#) and [target file \[Page 15\]](#) descriptions in the [commands for loading and unloading data \[Page 31\]](#). These rules are explained in more detail.

You can go from the description of the commands for loading and unloading data directly to the description of the rules used in each case.

### bool\_spec

#### Syntax

`<bool_spec> ::= BOOLEAN <valVALUE_FOR_TRUE>/<valVALUE_FOR_FALSE>`

<code>valVALUE_FOR_TRUE</code>	Defines the character string for values that are true
<code>valVALUE_FOR_FALSE</code>	Defines the character string for values that are false

The default value in the Replication Manager is `TRUE/FALSE`.

The character strings can contain a maximum of 10 characters.

#### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the character string that represents the BOOLEAN values in data files unloaded from a database instance or loaded into a database instance.

You can also change the current value for individual commands.

### code\_spec

#### Syntax

`<code_spec> ::= <standard code spec \[Page 81\]> | CODESET <valCODESET_NAME>`

<code>valCODESET_NAME</code>	Code attribute
------------------------------	----------------

#### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

By specifying the code attribute, you define the standard value used to interpret data files in [plain text \[Page 17\]](#).

### date\_spec

#### Syntax

`<date_spec> ::= DATE <standard date mask \[Page 81\]> | DATE '<valFREE_MASK>'`

<code>'&lt;valFREE_MASK&gt;'</code>	Output format
-------------------------------------	---------------

	Specify which of the three possible components day, month, and year you want to appear in the date. You can use either the German abbreviations <b>T</b> , <b>M</b> and <b>J</b> or the English abbreviations <b>D</b> , <b>M</b> and <b>Y</b> . If you enter three characters for the month, that is <b>MMM</b> , the name of the month is displayed as the standard abbreviation, for example, Oct for October.
--	---

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the format for [plain text values \[Page 17\]](#) in which DATE columns are entered and output.

This format only applies to the load or unload command in which it is specified. Otherwise, the Replication Manager default is used.

Use the [SET command \[Page 28\]](#) to view the Replication Manager default.

## delimiter\_spec

### Syntax

```
<delimiter_spec> ::= DELIMITER '<valDELIMITER>'
```

'<valDELIMITER>'	DELIMITER display, No character, or exactly one character long Representation of a blank DELIMITER by specification of blank single quotation marks ( ' ' )
------------------	--

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify which character is used in [COMPRESSED \[Page 18\]](#) format data files to select data. If you want to unselected data, enter a blank DELIMITER in the command.

Use the [SET command \[Page 28\]](#) to view the Replication Manager default.

## extract\_files\_spec

### Syntax

```
<extract_files_spec> ::= <data_outfile_spec> [Page 80] [<longfile_spec_mlt> [Page 78]]  
| <command_outfile_spec [Page 80]> <data_outfile_spec> [<longfile_spec_mlt>]
```

data_outfile_spec	Syntax rule for defining the <a href="#">target file [Page 15]</a> for data
command_outfile_spec	Syntax rule for defining the file for the generated load commands ( <a href="#">command file [Page 13]</a> )
longfile_spec_mlt	Target file for data from table fields defined as LONG values

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

The first specification of this rule only applies to [DATAEXTRACT commands \[Page 36\]](#) that do not create control files (variant 1 of the DATEXTRACT command).



```
DATAEXTRACT * FROM article
OUTFILE 'article.data'
```

The second specification only applies to DATAEXTRACT commands that enable a complete recovery of the table (variant 2 of the DATAEXTRACT command).



```
DATAEXTRACT FOR DATALOAD TABLE article
OUTFILE 'article.command'
OUTFILE 'article.data'
```

## file\_extract\_spec

### Syntax

```
<file_extract_spec> ::= START <valSTART_POS> [<valRECORD_COUNT>]
```

valSTART_POS	Number of the first data record to be loaded
valRECORD_COUNT	Number of data records to be loaded

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

When you load data ([commands for loading and unloading data \[Page 31\]](#)), you describe in a FASTLOAD or DATALOAD command in the [infile\\_spec \[Page 71\]](#) that only a certain section of the source file is to be loaded.

- When you specify **START <valSTART\_POS>**, the rest of the source file is loaded in full, starting with the specified data record.
- When you specify **START <valSTART\_POS> [<valRECORD\_COUNT>]**, <valRECORD\_COUNT> data records are loaded, starting with the first specified data record.

If the source files have the format **COMPRESSED** or **FORMATTED**, the REPM Server uses the number of the specified data record to determine the line in the source file as of which loading starts.

If the source files have the format **FORMATTED BINARY**, the REPM Server uses the number of the specified data record and the length of an individual data record in the source file to determine the position in the source file as of which loading starts.



Lines 5-15 of source file `customer.data` are to be loaded. As line 5 is to be included when the data is loaded, you must specify 11 as the number of lines.

```
FASTLOAD TABLE customer
  cno          1
  last name    2
  zip          3
  place        4
INFILE 'customer.data' START 5 11
```

## file\_format\_spec

### Syntax

```
<file_format_spec> ::= <code_spec [Page 68]> <file_format_spec>
| <number_spec [Page 68]> <file_format_spec>
| <date_spec [Page 79] <file_format_spec>
| <time_spec [Page 68] <file_format_spec>
| <timestamp_spec [Page 68] <file_format_spec>
| <null_spec [Page 83] <file_format_spec>
| <bool_spec [Page 83] <file_format_spec>
| <int_spec [Page 83]> <file_format_spec>
| <separator_spec [Page 83]> <file_format_spec>
| <delimiter_spec [Page 80]> <file_format_spec>
| FORMATTED [Page 80] <file_format_spec>
| COMPRESSED [Page 68] <file_format_spec>
| BINARY [Page 68] <file_format_spec>
```

### Use

This is a [syntax rule for describing data access \[Page 71\]](#).

Use it in a [command for loading or unloading data \[Page 71\]](#) to specify the format of the data and the data files ([data file formats \[Page 81\]](#)).

## infile\_spec

### Syntax

```
<infile_spec> ::= INFILE <'valFILE_NAME'> [<file_format_spec [Page 71]>]
[<file_extract_spec [Page 70]>] [<noheader_spec [Page 79]>]
```

'valFILE_NAME'	Name and path of <a href="#">source file [Page 14]</a>
----------------	--

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

When data is loaded ([command for loading and unloading data \[Page 31\]](#)), you describe the source file.

The rules `file_extract_spec` and `noheader_spec` are only evaluated for FASTLOAD and DATALOAD commands. If you use these rules in other commands, they are ignored.

- You can use `file_extract_spec` to specify that only certain parts of a file are to be loaded.
- For **binary** files, you can use `noheader_spec` to specify that the file does not have a special header containing the sizes of a data record.

## int\_spec

### Syntax

```
<int_spec> ::= INTEGER HILO | INTEGER LOHI
```

INTEGER HILO	The current data file stores integers so that the byte with the lowest valency is
--------------	---

	stored first (the big endian in the binary number)
<b>INTEGER LOHI</b>	The current data file stores integers so that the byte with the highest valency is stored first (little endian, byte swap)

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to define the representation of integers in data files. You can only do this for data files with the [FORMATTED BINARY \[Page 20\]](#) file format. This rule is ignored when you load or unload data with data files that have the [COMPRESSED \[Page 18\]](#) file format.

If the representation specified for a data file does not match the current host, the values are adjusted before being loaded into the database or written to the data file.

## longfile\_code\_spec

### Syntax

`<longfile_code_spec> ::= <code\_spec \[Page 68\]> | BINARY`

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

By specifying the code attribute, you define the standard value used to interpret data files in LONG data files.

## longfile\_spec

### Syntax

`<longfile_spec> ::= LONGFILE <valCOLUMN_ID> <longfile\_code\_spec \[Page 72\]>  
| LONGFILE <valCOLUMN_NAME> <longfile_code_spec>  
| LONGFILE <valCOLUMN_ID> <valFILE_NAME> [<longfile_code_spec>]  
| LONGFILE <valCOLUMN_NAME> <valFILE_NAME> [<longfile_code_spec>]`

<b>valCOLUMN_ID</b>	Column ID
<b>valCOLUMN_NAME</b>	Column name
<b>valFILE_NAME</b>	Name and path of data file

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

If you use one of the first two syntax rules, `longfile_code_spec` is a required specification. In all other cases, this specification is optional.

- The first two syntax rules relate to loading/unloading in cases where each LONG value is in a separate LONG data file.
- The other rules relate to loading/unloading in cases where all LONG values are in one LONG data file.

**See also:**

[Loading LONG Values \[Page 73\]](#)

[Unloading LONG Values \[Page 76\]](#)





## Loading LONG Values

### Use

In a [command for loading data \[Page 31\]](#) use the syntax rule [longfile\\_spec \[Page 72\]](#) in [longfile\\_spec\\_mlt \[Page 78\]](#) to specify for each column with the internal data type LONG the name and path of the data file from which you want to load the LONG values.

When you load LONG values, you can only use the **LONGFILE** <valCOLUMN\_NAME> '**<valFILE\_NAME>**' variant of the syntax rule, since you can only use column names in the load command, and no column IDs (see [column\\_descriptor \[Page 52\]](#)).

When you load LONG values, you must differentiate between the following cases:

- Case 1: Each LONG value to be inserted is in a separate data file ([Each LONG value to be inserted is in a separate data file \[Page 73\]](#)).
- Case 2: All of the LONG values to be inserted are in one data file ([All of the LONG values are in one data file \[Page 75\]](#)).



## Each LONG Value to Be Inserted in a Separate LONG Data File

When you [load LONG values \[Page 73\]](#), you must differentiate between the following cases:

- Case 1: Each LONG value of a column is in a separate LONG data file
- Case 2: All of the LONG values to be inserted are in one LONG data file ([All of the LONG values to be inserted are in one LONG data file \[Page 75\]](#)).

### Case 1: Each LONG value to be inserted for a column is in a separate LONG data file

You specify the names of the files that contain the LONG values in the [source file \[Page 14\]](#) instead of the column values (if necessary entering position specifications). The entire files or parts of them are loaded as LONG values.



Entire files are loaded as LONG values.

```
DATALOAD TABLE hotel
  cno      1
  name     2
  info     3
INFILE 'hotel.data'
```

Contents of the [source table \[Page 15\]](#):

```
10,Excelsior, 'EXCELSIOR.LNG'
30,Flora, 'FLORA.LNG'
60,Bellevue, 'BELLEVUE.LNG'
```



Parts of the files are loaded as LONG values.

```
DATALOAD TABLE hotel
  cno      1
  name     2
```

```

info      3
INFILE 'hotel.data'

```

Contents of the source table:

```

10,Excelsior,'EXCELSIOR.LNG' 1-880
30,Flora,'FLORA.LNG' 8-1046
60,Bellevue,'BELLEVUE.LNG' 100-260

```

## Specifying a Code Attribute

In the load command, it is possible to specify a code attribute for all files that contain the LONG values that are to be loaded into a column. This is an optional entry. You can only enter a single code type for all files. It is not possible to specify the code type for each individual file.

If the code attribute is not specified in the command, the Replication Manager uses a default value for the code type of the file. This is derived from the data type of the column to be loaded. The following conversion table is used:

Data type in the database	External code type
LONG ASCII	ASCII
LONG BYTE	BINARY
LONG UNICODE	UCS2

If the external code type and the internal database data type are different, the Replication Manager converts the data. If the types are incompatible, the Replication Manager generates an error message and stops processing the command.

### Compatibility table

External code type	Data type in the database	Compatible yes/no
ASCII	LONG ASCII LONG BYTE LONG UNICODE	yes yes yes
UCS2	LONG ASCII LONG BYTE LONG UNICODE	no yes yes
BINARY	LONG ASCII LONG BYTE LONG UNICODE	no no yes



### Specification of ASCII code attribute ASCII

```

DATALOAD TABLE hotel
  cno      1
  name     2
  info     3
INFILE 'hotel.daten'
LONGFILE info ASCII

```

Contents of the [source table \[Page 15\]](#):

```

10,Excelsior,'EXCELSIOR.LNG'
30,Flora,'FLORA.LNG'
60,Bellevue,'BELLEVUE.LNG'

```



No file name can be used after the keyword LONGFILE.



## All LONG Values to Be Inserted in One Data File

When you [load LONG values \[Page 73\]](#), you must differentiate between the following cases:

- Case 1: Each LONG value to be inserted is in a separate data file ([Each LONG value to be inserted is in a separate data file \[Page 73\]](#)).
- Case 2: All of the LONG values to be inserted for a column are in one file.

### Case 2: All of the LONG values to be inserted for a column are in one file.

#### Variant 1

In the [source file \[Page 14\]](#), you specify the name of the file containing the LONG values with the start and end position of the LONG value to be loaded, instead of a column value.



```
DATALOAD TABLE hotel
  cno      1
  name     2
  info     3
INFILE 'hotel.data'
```

Contents of the source file:

```
10,Excelsior,'HOTEL.LNG' 1-880
30,Flora,'HOTEL.LNG' 881-1046
60,Bellevue,'HOTEL.LNG' 1047-1360
```

#### Variant 2

You specify the name of the file containing the LONG values in the command after the keyword **LONGFILE** and not before the positions specified in the source file.



```
DATALOAD TABLE hotel
  cno      1
  name     2
  info     3
INFILE 'hotel.data'
LONGFILE info 'HOTEL.LNG'
```

Contents of the source file:

```
10,Excelsior,1-880
30,Flora,881-1046
60,Bellevue,1047-1360
```

#### Specification of a code attribute for the files specified with LONGFILE (variant 2)

In the load command, it is possible to specify the code attribute for files specified with LONGFILE. This is an optional entry.

If the code attribute is not specified in the command, the Replication Manager uses a default value for the code type of the file. This is derived from the data type of the column to be loaded. The following conversion table is used:

Data type in the database	External code type
LONG ASCII	ASCII

LONG BYTE	BINARY
LONG UNICODE	UCS2

If the external code type and the internal database data type are different, the Replication Manager converts the data. If the types are incompatible, the Replication Manager generates an error message and stops processing the command.

#### Compatibility table

External code type	Data type in the database	Compatible yes/no
ASCII	LONG ASCII	yes
	LONG BYTE	yes
	LONG UNICODE	yes
UCS2	LONG ASCII	no
	LONG BYTE	yes
	LONG UNICODE	yes
BINARY	LONG ASCII	no
	LONG BYTE	no
	LONG UNICODE	yes



#### Specification of ASCII code attribute ASCII

```

DATALOAD TABLE hotel
  cno      1
  name     2
  info     3
INFILE 'hotel.data'
LONGFILE info 'HOTEL.LNG' ASCII

```

Contents of the source file:

```

10,Excelsior,1-880
30,Flora,881-1046
60,Bellevue,1047-1360

```



## Unloading LONG Values

### Use

In a [command for unloading data \[Page 31\]](#) use the syntax rule [longfile\\_spec \[Page 72\]](#) in [longfile\\_spec\\_mlt \[Page 78\]](#) to specify for each column with the [internal database type \[Page 16\]](#) LONG, the name and path of the LONG data file into which you want to load the LONG values.

You can use both variants of the syntax rule to unload the data.

Always use a column ID for the LONG column that you want to unload if a column ID is specified in the column list for this column (see [output\\_column\\_list \[Page 63\]](#)).

Use the column name if the column name is specified in the column list.



```

DATAEXTRACT * from hotel
OUTFIELDS  cno      1
           name     2
           zip      3
           address  4
           info     5

```

```

OUTFILE 'hotel.data'
LONGFILE info 'info.data'

DATAEXTRACT cno, name FROM hotel
OUTFIELDS 1 1
          2 2
OUTFILE 'hotel.data'
LONGFILE 2 'info.data'

```

When you unload LONG values, you must differentiate between the following cases:

- Case 1: Each LONG value in a column to be unloaded is written to a separate LONG data file ([Each LONG value to be unloaded is in a separate LONG data file \[Page 77\]](#)).
- Case 2: All of the LONG values to be unloaded are in one LONG data file ([All of the LONG values to be unloaded are in one LONG data file \[Page 78\]](#)).



## Each LONG Value to Be Unloaded in a Separate LONG Data File

When you unload LONG values, you must differentiate between the following cases:

- Case 1: Each LONG value in a column is written to a separate LONG data file.
- Case 2: All of the LONG values to be unloaded are in one LONG data file ([All of the LONG values to be unloaded are in one LONG data file \[Page 78\]](#)).

### Case 1: Each LONG value in a column to be unloaded is written to a separate LONG data file.

In the command for unloading data, you specify the name of the LONG data file with a number of placeholders for sequentially generated LONG data files for each LONG column that you want to unload.

If the LONG column you want to unload has no value in a data record (the value is an empty character string), an empty LONG data file is generated for this LONG value.

The generated unique file name enables the individual LONG data files to be assigned to the corresponding data record in the target table.



Use a sufficient number of numeric characters at the end of the file name as a placeholder. If the upper limit is reached while the data is being unloaded, but there are still values left to be unloaded, the Replication Manager generates an error message and terminates the command.



```

DATAEXTRACT * FROM hotel
  cno      1
  name     2
  info     3
OUTFILE 'hotel.data' FORMATTED
LONGFILE info 'info.data.###'

```

The Replication Manager generates the file with the name `info.data.001` for the first LONG value to be unloaded, the file `info.data.002` for the second, and so on. Content of the [target table \[Page 15\]](#):

```
10,Excelsior,'info.daten.001'
30,Flora, 'info.daten.002'
60,Bellevue,'info.daten.003'
```



## All LONG Values to Be Unloaded Are in One File

When you unload LONG values, you must differentiate between the following cases:

- Case 1: Each LONG value in a column to be unloaded is written to a separate data file ([Each LONG value to be unloaded is in a separate data file \[Page 77\]](#)).
- Case 2: All LONG values in a column to be unloaded are written to one data file

### Case 2: All LONG values in a column to be unloaded are written to one file

In the command for unloading data you enter the name of a data file for each LONG column, into which the LONG values of this LONG column are entered.

If the LONG column you want to unload has no value in a data record (the value is an empty character string), the position for this LONG value is generated as follows: The start position is the end position of the preceding LONG value in the column plus 1; the end position is the end position of the preceding LONG value in the column. This means that the start position is always 1 larger than the end position.

The start and end position of the generated LONG value in the data file enable the LONG values to be assigned to the data records of the target file.



```
DATAEXTRACT * FROM hotel
  cno      1
  name     2
  info     3
OUTFILE 'hotel.data'
LONGFILE info 'info.data'
```

Contents of the target file:

```
10,Excelsior,1-880
30,Flora,881-1046
60,Bellevue,1047-1360
```



## longfile\_spec\_mlt

### Syntax

```
<longfile_spec_mlt> ::= <longfile\_spec> \[Page 72\] <longfile_spec_mlt>
```

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

You can use the [DATALOAD \[Page 34\]](#), [DATAEXTRACT \[Page 36\]](#), and [DATAUPDATE \[Page 37\]](#) commands to load LONG values to a [target table \[Page 15\]](#), unload them from a [source table \[Page 15\]](#), or modify the LONG values in a table. Use this rule to define the columns in a target table to which you want to load LONG values, or from which source table you want to unload them.

The data for LONG values is stored separately from the other data in LONG data files. Specify these files with the syntax rule [longfile\\_spec \[Page 72\]](#).

If, when you are unloading LONG values, you specify more LONG data files than there are LONG columns in the column list (see [output column list \[Page 63\]](#)), the Replication Manager ignores the surplus LONG data files.

**See also:**

[Loading LONG Values \[Page 73\]](#)

[Unloading LONG Values \[Page 76\]](#)

## noheader\_spec

### Syntax

**<noheader\_spec> ::= NOHEADER <valRECORD\_LENGTH>**

<b>valRECORD_LENGTH</b>	Length of an individual data record in the source file
-------------------------	--

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

When you load data ([Command for loading or unloading data \[Page 31\]](#)) with a FASTLOAD or DATALOAD command, you define in [infile\\_spec \[Page 71\]](#) that a file in FORMATTED BINARY format does not contain a special header. At the same time, you specify the length of an individual data records in the source file.

This is an optional entry.

If the source file was generated with the DATAEXTRACT command, it contains a special header with the length of an individual data record in the file.

If this header is missing, the REPM Server can use `noheader_spec` to specify the length of an individual data record.

If this rule is missing, the REPM Server calculates the length of an individual data records using the information on the column positions in the load command.

## number\_spec

### Syntax

**<number\_spec> ::= DECIMAL ' /<t>/<d>/'**

<b>&lt;t&gt;</b>	Defines the character for grouping thousands This value is optional; optional; optional; The default value in the Replication Manager is no character
<b>&lt;d&gt;</b>	Defines the character used to separate integers from decimal places; The default value in the Replication Manager is a period

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the characters used in decimal numbers to group thousands and separate integers from decimal places.

You can also change the current value for individual commands.

## null\_spec

### Syntax

`<null_spec> ::= NULL '<valLITERAL>'`

'<valLITERAL>'	Null value representation; maximum length of 20 characters  The default value in the Replication Manager is: ? (a question mark and 19 space characters).
----------------	--

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).


Use it to specify the character string used to represent null values, which were loaded from the database instance, in data files.

You can also change the current value for individual commands.

## outfile\_spec

### Syntax

`<outfile_spec> ::= OUTFILE '<valFILE_NAME>' <file\_format\_spec> \[Page 71\] [APPEND]`

'<valFILE_NAME>'	Name and path of the <a href="#">target file [Page 15]</a> In a <a href="#">DATAEXTRACT command [Page 36]</a> , also the name of the generated <a href="#">command file [Page 13]</a>
APPEND	An existing target file is not overwritten. This means that data or generated load statements are added to the end of the file.    This does <b>not</b> apply to TABLEEXTRACT commands. A TABLEEXTRACT command overwrites an existing target file.

### Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to describe the target file or command file when you [unload data \[Page 31\]](#).

## part\_spec

### Syntax

`<part_spec> ::= TABLE <table\_name \[Page 48\]> | USER | ALL`

TABLE	A single table is saved to a file, or recovered from a file. The user must be the owner of this table.
USER	The tables of the user currently logged on to the Replication Manager are saved to one or more files, or the tables unloaded with <b>TABLEEXTRACT USER</b> are recovered from this file or files. The user who recovers the tables does not need to be the user who saved them.  This means that tables unloaded by <code>user1</code> with <b>TABLEEXTRACT USER</b> can be reloaded by <code>user1</code> .



<b>ALL</b>	<p>All tables of the database instance are saved to one or more files or recovered from this file or files.</p> <p>The user must have the rights of the database administrator (SYSDBA).</p> <p>The users whose tables were saved must exist in the target database when the tables are recovered.</p>
------------	--

## Use

Use this [syntax rule for describing data access \[Page 68\]](#) in a [TABLEEXTRACT command \[Page 40\]](#) or [TABLELOAD command \[Page 38\]](#) to specify whether a single table, the tables of the logged on user, or all tables of the database instance are saved or recovered.

## separator\_spec

### Syntax

**<separator\_spec> ::= SEPARATOR '<valSEPARATOR>'**

'<valSEPARATOR>'	Separator for data fields; must be one character long
------------------	---

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify which character is used to separate data fields in data files that have the [COMPRESSED \[Page 18\]](#) format.

Use the [SET command \[Page 28\]](#) to view the Replication Manager default.

## standard\_code\_spec

**<standard\_code\_spec> ::= ASCII | EBCDIC | UCS2 | UTF8**

## Use

This is a [syntax rule for describing data access \[Page 68\]](#) (component of the rule [code\\_spec \[Page 68\]](#)).

By specifying the code attribute, you define the standard value used to interpret data files in [plain text \[Page 17\]](#).

You can also change the current value for individual commands.

The default value in the Replication Manager is ASCII.

## standard\_date\_mask

### Syntax

**<standard\_date\_mask> ::= ISO | USA | EUR | JIS | INTERNAL**

ISO	YYYY-MM-DD
USA	MM/DD/YYYY
EUR	DD.MM.YYYY

JIS	YYYY-MM-DD
INTERNAL	YYYYMMDD

Y stands for year, M for month, and D for day.

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the format for [plain text values \[Page 17\]](#) in which DATE columns are entered and output.

This format applies to both [commands for loading and unloading data \[Page 31\]](#) and [SQL statements \[Page 84\]](#).

The default value in the Replication Manager is INTERNAL.

## standard\_time\_mask

### Syntax

<standard\_time\_mask> ::= ISO | USA | EUR | JIS | INTERNAL

ISO	HH.MM.SS
USA	HH:MM AM (PM)
EUR	HH.MM.SS
JIS	HH:MM:SS
INTERNAL	HHHHMMSS

H stands for hours, M for minutes, and S for seconds.



You can use the standard formats, or define your own format. To do this, use **H** for hours, **M** for minutes, and **S** for seconds. Use two figures for minutes and seconds. You can choose any number of figures for **H**.

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the format for [plain text values \[Page 17\]](#) in which TIME columns are entered and output.

This format applies to both [commands for loading and unloading data \[Page 31\]](#) and [SQL statements \[Page 84\]](#).

You can also change the current value for individual commands.

The default value in the Replication Manager is INTERNAL.

## standard\_timestamp\_mask

### Syntax

<standard\_timestamp\_mask> ::= ISO | USA | EUR | JIS | INTERNAL

ISO	YYYY-MM-DD-HH.MM.SS.NNNNNN
-----	----------------------------

USA	ISO
EUR	ISO
JIS	ISO
INTERNAL	YYYYMMDDHHMMSSNNNNNN

H stands for hours, M for minutes, S for seconds, and N for milliseconds or microseconds.

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the format for [plain text values \[Page 17\]](#) in which TIMESTAMP columns are entered and output.

This format applies to both [commands for loading and unloading data \[Page 31\]](#) and [SQL statements \[Page 84\]](#).

The default value in the Replication Manager is INTERNAL.

## time\_spec

### Syntax

```
<time_spec> ::= TIME <standard time mask \[Page 82\]> | TIME '<valFREE_MASK>'
```

'<valFREE_MASK>'	Self-defined output format Use <b>H</b> for hours, <b>M</b> for minutes, and <b>S</b> for seconds. Use two figures for minutes and seconds. You can choose any number of figures for <b>H</b> .
------------------	--

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the format for [plain text values \[Page 17\]](#) in which TIME columns are entered and output.

This format only applies to the [command for loading or unloading data \[Page 31\]](#) in which it is specified. Otherwise, the Replication Manager default is used.

Use the [SET command \[Page 28\]](#) to view the Replication Manager default.

## timestamp\_spec

### Syntax

```
<timestamp_spec> ::= TIMESTAMP <standard timestamp mask \[Page 82\]>  
| TIMESTAMP '<valFREE_MASK>'
```

'<valFREE_MASK>'	Self-defined output format Use <b>H</b> for hours, <b>M</b> for minutes, and <b>S</b> for seconds. Use two figures for minutes and seconds. You can choose any number of figures for <b>H</b> .
------------------	--

## Use

This is a [syntax rule for describing data access \[Page 68\]](#).

Use it to specify the format for [plain text values \[Page 17\]](#) in which TIMESTAMP columns are entered and output.

This format only applies to the [command for loading or unloading data \[Page 31\]](#) in which it is specified. Otherwise, the Replication Manager default is used.

Use the [SET command \[Page 28\]](#) to view the Replication Manager default.



## SQL Statements

All SQL statements can be specified in a [command file \[Page 13\]](#) that you enter when you [call the Replication Manager \[Page 21\]](#). Database queries, however, only produce status messages.

COMMIT and ROLLBACK statements are only active in the mode AUTOCOMMIT OFF ([AUTOCOMMIT Command \[Page 27\]](#)).

**See also:** *Reference Manual: SAP DB 7.2 and 7.3*



## Keywords and Alternative Keywords

Keywords are components of commands. They define the function of the command, which is modified by arguments and options.



```

DATALOAD TABLE article
    foa          01-08 CHAR
    des          09-39 CHAR
    stock        40-43 INTEGER
    min_order    44-45 INTEGER
    ordered      46-49 INTEGER
    del_date     50-57 CHAR
    price        58-65 DECIMAL (2)
    weight       66-69 REAL
INFILE 'customer.data' FORMATTED
  
```

The Replication Manager uses the keywords listed below. In some cases, you can also use the specified alternatives to the keywords.

[Keywords, Starting with A - C \[Page 84\]](#)

[Keywords, Starting with D - E \[Page 85\]](#)

[Keywords, Starting with F - K \[Page 86\]](#)

[Keywords, Starting with L - P \[Page 86\]](#)

[Keywords, Starting with R - S \[Page 87\]](#)

[Keywords, Starting with T - Z \[Page 87\]](#)



## Keywords A - C

The Replication Manager uses the following [keywords and alternative keywords \[Page 84\]](#):

Keyword	Alternative Keyword
<b>ALL</b>	
<b>AND</b>	
<b>ANSI</b>	
<b>APPEND</b>	

ASCII	
AUTOCOMMIT	
BINARY	
BOOLEAN	
BY	
CATALOGEXTRACT	EXTRACT CATALOG
CATALOGLOAD	LOAD CATALOG
CHAR	
CODESET	
CODETYPE	
COMPRESS	
COUNT	
CURRENT	



## Keywords D - E

The Replication Manager uses the following [keywords and alternative keywords \[Page 84\]](#):

Keyword	Alternative Keyword
DATAEXTRACT	EXTRACT DATA
DATALOAD	LOAD DATA
DATAUPDATE	UPDATE DATA
DATE	
DB2	
DBEXTRACT	EXTRACT DB
DBLOAD	LOAD DB
DEC	
DECIMAL	
DEFAULT	
DELIMITER	
DUPLICATES	
EBCDIC	
EUR	
EXTRACT CATALOG	CATALOGEXTRACT
EXTRACT DATA	DATAEXTRACT
EXTRACT DB	DB EXTRACT
EXTRACT TABLE	TABLE UNLOAD



## Keywords F - K

The Replication Manager uses the following [keywords \[Page 84\]](#):

Keyword
FASTLOAD
FOR
FORMATTED
HEX
HILO
IF
IGNORE
INFILE
INSTALLATION
INTEGER
INTERNAL
ISO
JIS
KEY



## Keywords L - P

The Replication Manager uses the following [keywords and alternative keywords \[Page 84\]](#):

Keyword	Alternative Keyword
LANGUAGE	
LOAD CATALOG	CATALOGLOAD
LOAD DATA	DATALOAD
LOAD DB	DBLOAD
LOAD TABLE	TABLELOAD
LOHI	
LONGFILE	
MESSAGE	
NOT	
NULL	
NUMBER	
OFF	
ON	
OR	
ORACLE	
ORDER	

OTHERWISE	
OUTFIELDS	
OUTFILE	
POS	



## Keywords R - S

The Replication Manager uses the following [keywords \[Page 84\]](#):

Keyword
REAL
REJECT
RELEASE
ROUND
ROWS
SCALE
SEPARATOR
SEQNO
SERVERDB
SET
SQLID
SQLMODE
STAMP
SYSDATE



## Keywords T - Z

The Replication Manager uses the following [keywords and alternative keywords \[Page 84\]](#):

Keyword	Alternative Keyword
TABLE	
TABLEEXTRACT	
TABLELOAD	LOAD TABLE
TABLEUNLOAD	EXTRACT TABLE
TABLEUPDATE	UPDATE TABLE
TERMCHARSET	
TIME	
TIMESTAMP	
TRUNC	
UID	

UPDATE	
UPDATE DATA	DATAUPDATE
UPDATE TABLE	TABLEUPDATE
USA	
USAGE	
USE	
USER	
USERGROUP	
USERKEY	
VERSION	
WITH	
ZONED	



## Processing Commands

The Replication Manager processes a sequence of commands and statements received in the form of a [command file \[Page 13\]](#) when the [Replication Manager is called \[Page 21\]](#). You create this command file and then use one of three methods to forward it to the REPM Server:

- By [using the Replication Manager CLI client program \(REPMCLI\) \[Page 88\]](#)
- By [using the Perl script language \[Page 89\]](#)
- By [using the Python script language \[Page 96\]](#)

The documentation only contains an extract of the script language functions used in the Replication Manager. For more information, see the vendor documentation for Perl and Python.



## Processing Commands with the Replication Manager CLI

### Syntax

**repmcli** [-u <userid>, <password> \[Page 26\]](#) [-d <database\\_name> \[Page 23\]](#) [-b <command\\_file> \[Page 22\]](#) [-E <number> \[Page 23\]](#)

<b>&lt;userid&gt;</b>	User name
<b>&lt;password&gt;</b>	User password
<b>&lt;database_name&gt;</b>	Name of database instance
<b>&lt;command_file&gt;</b>	Name of the <a href="#">command file [Page 13]</a>



By specifying the **-E** option, you define the number of errors that must occur before the execution of the command file is terminated.

You can also specify other [options \[Page 22\]](#).



## Use

You create a command file `<command_file>` in the form of simple text file and transfer the name of this file with the `-b` option when the [REPM Server \[Page 13\]](#) session is started ([Calling the Replication Manager \[Page 21\]](#)).

The Replication Manager processes the commands and statements contained in the command file. If errors (return code  $\neq 0$ ) occur, the processing of the command file is terminated as soon as a predefined number of errors is reached.

Error messages of this kind can be caused by several different things. Here are some examples:

- An SQL statement or [REPM Server command \[Page 26\]](#) can contain syntax errors
- The result of an SQL statement can be  $\neq 0$ . (for example, 100: no lines found)
- A line is refused when a command for loading data is executed



To enable you to respond to script errors, the Replication Manager functions are available as a library for the Perl and Python script languages. This means that you can use the functions of the Replication manager [in conjunction with Perl \[Page 89\]](#) and [Python \[Page 96\]](#).



## Using Perl

### Syntax

```
perl <perl_script_file> [<arguments> ...]
```



```
perl <perl_script_file> <userid> <password> <data_path>
```

### Prerequisites

Perl is installed on the host.

The following files are shipped with the Replication Manager program for processing Perl scripts:

- NT: repmanperl.dll, repman.pm, instperl.pl
- UNIX: repmanperl.so, repman.pm, instperl.pl
- HP: repmanperl.sl, repman.pm, instperl.pl



Open the `instperl.pl` file. When you do this, the files are copied to the relevant directories.

The Replication Manager supports Perl integration from Perl Version 5.005\_02 onwards.

### Template for Batch Files

The following section contains examples of batch files in Perl. Their equivalents in the [Python \[Page 96\]](#) script language are also provided for comparison.

Example no. 1 Build a Perl module with reference to the SAP DB Perl Libraries, parse the call arguments	<a href="#">Perl [Page 91]</a>	<a href="#">Python [Page 99]</a>
---	--------------------------------	----------------------------------

Example. 2 Set up a user session with the REPM Server, Log onto the database instance, Log off	<a href="#">Perl</a> <a href="#">[Page 92]</a>	<a href="#">Python</a> <a href="#">[Page 100]</a>
Example no. 3 Start a user session, Log onto the database instance, Query error code to determine whether table exists Create a table without querying the error code Log off	<a href="#">Perl</a> <a href="#">[Page 93]</a>	<a href="#">Python</a> <a href="#">[Page 101]</a>
Example no. 4 Start a user session Log onto the database instance Create a table and query the error code Load data to the table using REPM Server command(s) and query the error code Log off	<a href="#">Perl</a> <a href="#">[Page 94]</a>	<a href="#">Python</a> <a href="#">[Page 102]</a>
Example no. 5 Start a user session Log onto the database instance Load data to the table using REPM Server command(s) and intercept exceptions Log off	<a href="#">Perl</a> <a href="#">[Page 95]</a>	<a href="#">Python</a> <a href="#">[Page 104]</a>

[Perl Classes \[Page 90\]](#)



## Perl Classes

You use the following classes to create batch files for the Replication Manager with the Perl script language:

[RepMan Class \[Page 90\]](#)

[Exception Classes \[Page 91\]](#)



## Perl: RepMan Class

### Constructor: RepMan (<server\_node>, <database\_name>)

#### Brief Description

Creates a connection to the [REPM Server \[Page 13\]](#)

If the server node name and name of the database instance are specified, the system assumes that the database instance, data, and REPM Server are located on a remote server.

If only the name of the database instance is specified, the system assumes that the database instance, data, and REPM Server are located on the local server. The suitable REPM Server is determined from the release of the specified database instance.

If neither the server node name nor the name of the database instance is specified, the system establishes a connection to the newest REPM Server on the local system.

The session is closed again when the object is deleted with `undef $session`.



```
$session = RepMan ('p12345', 'mydb')
```

## Method: cmd (<commandstring>)

### Brief Description

An [SQL statement \[Page 84\]](#) or [REPM Server command \[Page 26\]](#) is executed.

The script is terminated if the command fails.



```
$output = $session->cmd ("DATALOAD TABLE customer".  
  
    "cno      1-4".  
    "surname  6-12".  
    "zip      14-18".  
    "place    20-31".  
    "INFILE '$data_path\customer.dat' ")
```

## Method: sql (<commandstring>)

### Brief Description

An SQL or REPM Server command is executed.

Script execution is terminated when an REPM Server command fails.

A return code is output when an SQL statement fails.



```
$result = $session->sql ('EXISTS TABLE MYTABLE')
```



## Perl: Exception Classes

An exception is a string beginning with either `CommunicationError` or `RepManServerError`.



```
RepManServerError: 25011 SQL error -3005 = Invalid SQL Statement
```

Example of use in a script:

```
eval {$session->cmd ('complete nonsense;')} ;  
if ($?) { print "command failed: $@\n"; }
```

Print output:

```
command failed: RepManServerError: 25011 SQL error -3005 = Invalid  
SQL Statement
```



## Perl: Example No. 1

Build a Perl module with reference to the SAP DB Perl Libraries,  
parse the call arguments:

```
# Reference to SAP DB Perl Library
```

```
# -----
use SAP::DBTECH::repman;

# Parse the call arguments
# -----
$user = $ARGV[0];
$pwd = $ARGV[1];
$dbname = $ARGV[2];
$data_path = $ARGV[3];
$host = "localhost";
```



## Perl: Example No. 2

Set up a user session with the [REPMServer \[Page 13\]](#)  
 Log on to the database instance  
 Log off

```
# Reference to SAP DB Perl Library
# -----
use SAP::DBTECH::repman;

# Parse the call arguments
# -----
$user = $ARGV[0];
$pwd = $ARGV[1];
$dbname = $ARGV[2];
$data_path = $ARGV[3];
$host = "localhost";

# Start a user session with the REPMServer
# -----
$session = repman::RepMan ($host, $dbname);

# Log on to the database instance
# -----
$session->cmd("use user $user $pwd;");

# Log off
# -----
undef $session
```



## Perl: Example No. 3

Set up a user session  
 Log on to the database instance  
 Query whether table exists by querying the error code  
 Create a table without querying error code  
 Log off

```
# Reference to SAP DB Perl Library
# -----
use SAP::DBTECH::repman;

# Parse the call arguments
# -----
$user = $ARGV[0];
$pwd = $ARGV[1];
$dbname = $ARGV[2];
$data_path = $ARGV[3];
$host = "localhost";

# Start a user session with the REPMServer
# -----
$session = repman::RepMan ($host, $dbname);

# Log on to the database instance
# -----
$session->cmd("use user $user $pwd;");

# Query error code to determine whether table exists
# The sql method is used for this purpose
# -----
$rc = $session->sql('EXISTS TABLE CUSTOMER')

If $rc!=0
# Create the table CUSTOMER
# -----
$session->cmd ( 'CREATE TABLE customer ( '.
               'cno                FIXED(4) , '.
               'surname            CHAR(10) ASCII, '.
               'zip                CHAR(5)  ASCII, '.
               'place              CHAR(12) ASCII, '.
```

```
'PRIMARY KEY (cno) ')
```

```
# End the database session
```

```
# -----
```

```
undef $session
```



## Perl: Example No. 4

Set up a user session

Log on to the database instance

Create a table and query error code

Use [REPM\\$Server command\(s\) \[Page 26\]](#) to load data into table and query error code

Log off

```
# Reference to SAP DB Perl Library
```

```
# -----
```

```
use SAP::DBTECH::repman;
```

```
# Parse the call arguments
```

```
# -----
```

```
$user = $ARGV[0];
```

```
$pwd = $ARGV[1];
```

```
$dbname = $ARGV[2];
```

```
$data_path = $ARGV[3];
```

```
$host = "localhost";
```

```
# Start a user session with the REPMServer
```

```
# -----
```

```
$session = repman::RepMan ($host, $dbname);
```

```
# Log on to the database instance
```

```
# -----
```

```
$session->cmd("use user $user $pwd;");
```

```
$rc = $session->sql('EXISTS TABLE CUSTOMER')
```

```
If $rc!=0
```

```
# Create the table CUSTOMER
```

```
# -----
```

```
$session->cmd ( 'CREATE TABLE customer ( '.
```

```
'cno
```

```
FIXED(4), '.
```

```

'surname      CHAR(10) ASCII, '.
'zip          CHAR(5)  ASCII, '.
'place        CHAR(12) ASCII, '.
'PRIMARY KEY (cno) ')

print $rc

If $rc==0
    # Then branch of the If statement must be indented in Python
    # Load table CUSTOMER
    # -----
    $loadrc = $session->cmd ("DATALOAD TABLE customer ".
                            "cno      1-4".
                            "surname  6-12".
                            "zip      14-18".
                            "place    20-31".
                            "INFILE $data_path\customer.dat" )

    print $loadrc

# End the database session
# -----
undef $session

```



## Perl: Example No. 5



If a command is changed at runtime, for example, because user inputs are part of the command, syntax errors can easily occur.

The SQL method only provides SQL error codes. You need exceptions to intercept syntax errors in the [DATALOAD command \[Page 34\]](#).

Set up a user session

Log on to the database instance

Use [REPMServer command\(s\) \[Page 26\]](#) to load data into table and intercept exceptions

Log off

An incorrect data path in the following example can lead to a syntax error or data access error.

```

# Reference to SAP DB Perl Library
# -----
use SAP::DBTECH::repman;

# Parse the call arguments

```

```

# -----
$user = $ARGV[0];
$pwd = $ARGV[1];
$dbname = $ARGV[2];
$data_path = $ARGV[3];
$host = "localhost";

# Start a user session with the REPMServer
# -----
$session = repman::RepMan ($host, $dbname);

# Log on to the database instance
# -----
$session->cmd("use user $user $pwd;");

# Example of exception handling
# -----
eval{
    $loadrc=$session->sql ("DATALOAD TABLE customer ".
                                "cno          1-4".
                                "surname      6-12".
                                "zip          14-18".
                                "place        20-
31".
                                "INFILE $data_path\customer.dat" );
    print "$loadrc\n";}
if ($@){
    print "command failed: $@\n";
}

# End the database session
# -----
undef $session

```



## Using Python

### Syntax

```
x_python <python_script_file> [<arguments>...]
```





Possible call arguments:

```
x_python <python_script_file> <userid> <password> <dbname>
<data_path>
```

## Prerequisites

The following Python modules are shipped with the Replication Manager program and enable you to write Python scripts that can be processed with the [REPM Server \[Page 13\]](#).

- NT: repman.pyd
- UNIX: repmanmodule.so
- HP: repmanmodule.sl



These are the modules you need to use the REPM Server functions. You do not need a complete PYTHON installation.

If you already have a PYTHON installation, add %INSTROOT%\misc to the PYTHONPATH variable.

The Replication Manager supports PYTHON as of version 1.5.2.

## Template for Batch Files

The following section contains examples of batch files in Python. Their equivalents in the [Perl \[Page 89\]](#) script language are also provided for comparison.

Example no. 1 Build a Python module with reference to the SAP DB Python Libraries, parse the call arguments	<a href="#">Python [Page 99]</a>	<a href="#">Perl [Page 91]</a>
Example. 2 Set up a user session with the REPM Server, Log onto the database instance, Log off	<a href="#">Python [Page 100]</a>	<a href="#">Perl [Page 92]</a>
Example no. 3 Set up a user session, Log onto the database instance, Query error code to determine whether table exists Create a table without querying the error code Log off	<a href="#">Python [Page 101]</a>	<a href="#">Perl [Page 93]</a>
Example no. 4 Start a user session Log onto the database instance Create a table and query the error code Load data to the table using <a href="#">REPM Server command [Page 26](s)</a> and query the error code Log off	<a href="#">Python [Page 102]</a>	<a href="#">Perl [Page 94]</a>
Example no. 5 Start a user session Log onto the database instance Load data to the table using REPM Server command(s) and intercept exceptions Log off	<a href="#">Python [Page 104]</a>	<a href="#">Perl [Page 95]</a>

[Python Classes \[Page 98\]](#)



## Python Classes

You use the following classes to create batch files for the Replication Manager with the Python script language:

[RepMan Class \[Page 98\]](#)

[Exception Classes \[Page 99\]](#)



## Python: RepMan Class

### Constructor: RepMan (<server\_node>, <database\_name>)

<server_node>	Server node name
<database_name>	Name of database instance

#### Brief Description

Creates a connection to the [REPM Server \[Page 13\]](#)

If the server node name and name of the database instance are specified, the system assumes that the database instance, data, and REPM Server are located on a remote server.

If only the name of the database instance is specified, the system assumes that the database instance, data, and REPM Server are located on the local server. The suitable REPM Server is determined from the release of the specified database instance.

If neither the server node name nor the name of the database instance is specified, the system establishes a connection to the newest REPM Server on the local system.

The session is closed again when the object is deleted with `del session`.



```
session = repman.RepMan ('p12345', 'mydb')
```

### Method: cmd (<commandstring>)

#### Brief Description

An [SQL statement \[Page 84\]](#) or [REPM Server command \[Page 26\]](#) is executed.

The script is terminated if the command fails.



```
output = session.cmd ("""DATALOAD TABLE customer
                        cno      1-4
                        surname  6-12
                        zip      14-18
                        place    20-31
                        INFILE '%s\customer.dat' """ %data_path)
```

### Method: sql (<commandstring>)

#### Brief Description

An SQL or REPM Server command is executed.

Script execution is terminated when an REPM Server command fails.

A return code is output when an SQL statement fails.



```
result = session.sql ('EXISTS TABLE MYTABLE')
```



## Python: Exception Classes

### Class: CommunicationError

#### Brief Description

No connection could be made to the [REPM Server \[Page 13\]](#).

The value of the exception is an instance with the following attributes:

- `errorCode`
- `message`

### Class: RepManServerError

#### Brief Description

The command has failed.

The value of the exception is an instance with the following attributes:

- `errorCode` (REPM Server error number)
- `message` (REPM Server error text)
- `sqlCode` (SQL error number if an SQL command fails, otherwise 0)
- `sqlMessage` (SQL error text if an SQL command fails, otherwise 0)



## Python: Example No. 1

Build a Python module with reference to the SAP DB Python Libraries, parse the call arguments:

```
# Reference to Python Libraries
# -----
import sys
import repman

# Parse the call arguments
# -----
user = sys.argv [1]
pwd = sys.argv [2]
dbname = sys.argv [3]
data_path = sys.argv[4]
```

```
host = ''
```



## Python: Example No. 2

Set up a user session with the [REPMServer \[Page 13\]](#)

Log on to the database instance

Log off

```
# Reference to Python Libraries
# -----

import sys
import repman

# Parse the call arguments
# -----

user = sys.argv [1]
pwd = sys.argv [2]
dbname = sys.argv [3]
data_path = sys.argv[4]
host = ''

# Connect to Replication Manager
# A new instance of the repman object is created
# Host determines the location of the REPMServer
# -----

session = repman.RepMan (host, dbname)

# Connect to database
# The cmd method is used for this purpose
# -----

session.cmd ('use user %s %s;' % (user, pwd))

# Log off by releasing the instance
# -----

del session
```



## Python: Example No. 3

Set up a user session  
 Log on to the database instance  
 Query whether table exists by querying the error code  
 Create a table without querying error code  
 Log off

```
# Reference to Python Libraries
# -----

import sys
import repman

# Parse the call arguments
# -----

user = sys.argv [1]
pwd = sys.argv [2]
dbname = sys.argv [3]
data_path = sys.argv[4]
host = ''

# Connect to Replication Manager
# A new instance of the repman object is created
# Host determines the location of the REPMServer
# -----

session = repman.RepMan (host, dbname)

# Connect to database
# The cmd method is used for this purpose
# -----

session.cmd ('use user %s %s;' % (user, pwd))

# Query error code to determine whether table exists
# The sql method is used for this purpose
# -----

rc = session.sql("EXISTS TABLE CUSTOMER")

If rc!=0
    # Then branch of the If statement must be indented in Python
    # Create the table CUSTOMER
    # -----
    session.cmd ( """CREATE TABLE CUSTOMER (
```

```

CNO                FIXED(4) ,
SURNAME            CHAR(10) ASCII,
ZIP                CHAR(5)  ASCII,
PLACE              CHAR(12) ASCII,
PRIMARY KEY (CNO) """)

```

```
session.cmd ("COMMIT")
```

```
# Log off by releasing the instance
```

```
# -----
```

```
del session
```



## Python: Example No. 4

Set up a user session

Log on to the database instance

Create a table and query error code

Use [REPMServer command\(s\) \[Page 26\]](#) to load data into table and query error code

Log off

```
# Reference to Python Libraries
```

```
# -----
```

```
import sys
```

```
import repman
```

```
# Parse the call arguments
```

```
# -----
```

```
user = sys.argv [1]
```

```
pwd = sys.argv [2]
```

```
dbname = sys.argv [3]
```

```
data_path = sys.argv[4]
```

```
host = ''
```

```
# Connect to Replication Manager
```

```
# A new instance of the repman object is created
```

```
# Host determines the location of the REPMServer
```

```
# -----
```

```
session = repman.RepMan (host, dbname)
```

```
# Connect to database
```

```
# The cmd method is used for this purpose
```

```

# -----
session.cmd ('use user %s %s;' % (user, pwd))

# Query error code to determine whether table exists
# The sql method is used for this purpose
# -----
rc = session.sql("EXISTS TABLE CUSTOMER")

If rc!=0
    # Then branch of the If statement must be indented in Python
    # Create the table CUSTOMER
    # -----
    rc = session.sql( """CREATE TABLE customer (
                                cno          FIXED(4) ,
                                surname      CHAR(10) ASCII,
                                zip          CHAR(5)  ASCII,
                                place       CHAR(12) ASCII,
                                PRIMARY KEY (cno) """ )

print rc

If rc==0
    # Then branch of the If statement must be indented in Python
    # Load table CUSTOMER
    # -----
    loadrc = session.sql ("""DATALOAD TABLE customer
                                cno          1-4
                                surname      6-12
                                zip          14-18
                                place       20-31
                                INFILE %s\customer.dat""" %data_path )

    print loadrc

session.cmd ("COMMIT")

# Log off by releasing the instance
# -----
del session

```



## Python: Example No. 5



If a command is changed at runtime, for example, because user inputs are part of the command, syntax errors can easily occur.

The SQL method only provides SQL error codes. You need exceptions to intercept syntax errors or other REPMServer errors in the [REPMServer commands \[Page 26\]](#).

Set up a user session

Log on to the database instance

Use REPMServer command(s) to load data into table and intercept exceptions

Log off

```
# Reference to Python Libraries
# -----

import sys
import repman

# Parse the call arguments
# -----

user = sys.argv [1]
pwd = sys.argv [2]
dbname = sys.argv [3]
data_path = sys.argv[4]
host = ''

# Connect to Replication Manager
# A new instance of the repman object is created
# Host determines the location of the REPMServer
# -----

session = repman.RepMan (host, dbname)

# Connect to database
# The cmd method is used for this purpose
# -----

session.cmd ('use user %s %s;' % (user, pwd))

# Example of exception handling
# -----

try:
    loadrc = session.sql ("""DATALOAD TABLE customer
                                cno          1-4
                                surname      6-12
                                zip          14-18
                                place        20-31
```



```
        INFILE %s\customer.dat"" %data_path )

    print "'%s'" % loadrc
except repman.RepManServerError, err:
    print 'command failed:', err

# Log off by releasing the instance
# -----
del session
```