Using SQL Server Database Compression with SAP NetWeaver

Applies to:

All SAP products based on SAP NetWeaver Application Server ABAP 7.0 and higher running on Microsoft SQL Server 2008 and higher. SAP NetWeaver Application Server Java does not currently support SQL Server database compression.

Summary

Database compression dramatically reduces the disk space needed for an SAP NetWeaver system running on Microsoft SQL Server. In particular, SAP Unicode systems benefit greatly from database compression. Space savings of 75% and higher have been observed on customer systems. This document describes the different SQL Server compression types and how to implement compression for an SAP NetWeaver system.

Customer feedback regarding PAGE compression was always positive. Therefore, SAP changed the default compression type from ROW to PAGE compression in May 2011.

Author: Martin Merdes

Company: Microsoft

Updated on: May 2011

Author Bio

Martin Merdes is a Senior Software Development Engineer at Microsoft. He has been working in different functions for Microsoft at SAP Germany since 1996. He is currently responsible for the SAP BW porting of Microsoft SQL Server. He contributed considerably to the development and testing of the database compression support in SAP NetWeaver. Furthermore, he is the developer of the compression report MSSCOMPRESS.

Table of Contents

Database Compression at a Glance

Depending on the SAP and SQL Server release, there are different compression types supported by SAP NetWeaver. Unfortunately the naming convention used by SAP and Microsoft is not consistent, which caused confusion in the past. In SAP notes the following compression types are mentioned: database compression, data compression, row compression, page compression, UCS-2 compression, Unicode compression, index compression, and vardecimal storage format (aka decimal compression or vardecimal data type).

SQL Server Compression Types

For any type of compression, Microsoft SQL Server currently requires the Enterprise or Datacenter Edition. The Developer Edition also works well, but is not supported by SAP and Microsoft for production usage. The following SQL Server releases introduced new compression features:

- SQL Server 2005 SP2
  In SQL Server 2005 SP2, the **vardecimal storage format** was released. This is a subset of the row compression type, which was released in SQL Server 2008. To use the vardecimal storage format for a single table, you have to enable the feature on database level first.

Note: Vardecimal storage format is deprecated as of SQL Server 2008. Microsoft no longer recommends using vardecimal storage format. Use row or page compression instead.

- SQL Server 2008
  A new feature called **data compression** was released in SQL Server 2008. Each table or index has one of the following compression types:
  - NONE: The table or index is not compressed. The rows are stored exactly the same way as in SQL Server 2005 and previous releases.
  - ROW: All rows are stored in variable length, including any numeric and fixed length character data types. **Row compression** does not cause measurable more on CPU consumption, but it saves disk space and reduces disk I/O. Therefore, always use compression type ROW rather than NONE.
  - PAGE: **Page compression** is always performed on top of row compression. It even saves more disk space, but there might be a higher CPU load on the database server. However, customer experience showed an improved overall system performance with PAGE compression due to disk I/O reduction and higher cache hit ratio. Furthermore, backup and restore is faster because of the smaller volume to be backed up.

  You can compress the data (heap, clustered index) and each (non-clustered) index separately. You can even choose a different compression type for each partition of an index. However, it is doubtful whether there is any benefit in doing so. Therefore, we do not recommend this. Row and page compression have no impact on LOB fields [data types: text, ntext, image, varchar(max), nvarchar(max), varbinary(max)].
- SQL Server 2008 R2
  SQL Server uses UCS-2 (not UTF-8) to encode Unicode characters. SQL Server 2008 R2 stores row- or page-compressed UCS-2 characters more efficiently than SQL Server 2008. The space usage of a UCS-2 character is roughly the same as that of a UTF-8 character, once a table is row- or page-compressed. This feature is called UCS-2 compression or simply **Unicode compression**.

Note: Unicode compression is not a new compression type. It is simply an improved algorithm to store UCS-2 characters when using row/page compression. After upgrading from SQL Server 2008 to SQL Server 2008 R2, all Unicode strings in newly inserted or updated rows are stored more efficiently. Existing Unicode strings in other rows are not changed. To fully benefit from Unicode compression for the whole database, you have to re-compress all tables. For example, apply row compression on the already row-compressed tables.
You can use the SAP report MSSCOMPRESS for this purpose.

Compression Types Supported by SAP

Depending on the SAP release, the following compression types are supported for an ABAP application server. For a Java application server there is currently no compression support.

- Vardecimal storage format
  The vardecimal storage format was used by SAP only for BW fact tables. The indexes of BW fact tables never benefited from it. As of SQL Server 2008, there is no longer any difference between BW fact tables and other SAP tables. By default, all tables are created using row compression. The vardecimal storage format is no longer used.

Note: The vardecimal storage format is a subset of the SQL Server 2008 row compression type. There is no advantage of the vardecimal storage format compared to row compression: in fact, the opposite is true. Therefore, SAP no longer recommends using the vardecimal storage format. Upgrade to SQL Server 2008 (R2) and use row or page compression instead.

- Row compression for tables (heap or clustered index)
  Before September 2010, the SAP Data Dictionary created new tables on SQL Server 2008 with compression type ROW and all new non-clustered indexes with compression type NONE. When converting an existing table manually in SAP transaction SE14 or during an SAP upgrade, the compression type of a table was not changed (it stayed NONE, ROW, or PAGE). However, non-clustered indexes were always created with compression type NONE.
- Row compression for (non-clustered) indexes
  In September 2010, SAP released correction instructions in SAP note 1459005 for SAP Basis 7.00 and all newer releases. These corrections are included in future SAP BASIS support packages. The corrected SAP data dictionary supports row and page compression for tables (with row compression as default). All new indexes are created with the same compression type as their table. SAP calls the new feature to compress indexes (as row- or page-compressed) **index compression**. This is actually not a SQL Server compression type. It is simply the name of an SAP feature.
- Page compression for tables and indexes
  In May 2011, SAP changed the default compression for new installations to page compression.

Applying SQL Server Data Compression for SAP

To fully benefit from row or page compression you have to perform the following steps:

1. Upgrade to SQL Server 2008 or higher
   For more information, see SAP note 1152240 *Setting Up Microsoft SQL Server 2008 (R2)*.
2. Update the SAP Data Dictionary
   Install the SAP_BASIS support packages or correction instructions as described in SAP note 1459005 *Enabling Index Compression for SQL Server*.
3. Configure SQL Server data compression within SAP
   Set a SAP profile parameter to set the default compression type of the SAP Data Dictionary, for example, to page compression.
4. Compress existing tables
   Use SAP report MSSCOMPRESS to compress existing tables using the same compression type as configured in the last step. Since the compression of a huge SAP database can take several days, you can perform the compression in chunks using SAP batch jobs.

Note: Compressing huge tables results in high resource consumption (CPU, I/O, blocking database locks). Therefore, do not run the compression during peak hours. The normal operation of your system might be affected by the compression jobs, even when using the *ONLINE* option. For more information about the *ONLINE* option, see SAP note 1006887.
Make sure that there is sufficient free space in the data and log files of the SAP- and tempdb-database. It is recommended to use the SQL Server recovery model *Bulk-logged* during compression to minimize the transaction log space needed. However, there are specific guidelines and recommendations when using SQL Server Database Mirroring. For more information, see SAP notes 965908 and 1550337.

Expected Compression Ratio

The amount of disk-space savings caused by data compression depends on many factors:

1. The original compression type
   An SAP system originally installed on SQL Server 2005, does not have any compressed tables. If SAP was installed on SQL Server 2008, it typically has row-compressed data (clustered index) and uncompressed (non-clustered) indexes.
2. The final compression type
   Ideally, you compress all tables (data and index) exactly the same way. Therefore, there is only one decision left: whether to use row or page compression. For SAP Unicode systems, we strongly recommend using SQL Server 2008 R2 (or higher) to benefit from Unicode compression.
3. State of index fragmentation
   During data compression, all indexes are completely reorganized. As a result, all data and index pages are filled completely – there is no free space within a database page. Therefore, you can make additional space savings from compression if the database has never before been reorganized.
4. Kind of data (distribution)
   Depending on the actual data, we have seen extreme compression rates for particular tables. In the worst case, there is only marginal compression, when most of the data is stored in LOB fields (for example in table BALDAT). In the best case, we have seen up to 90% space savings on particular customer tables (using page compression for data and index).

Although the compression ratio can vary, we have seen similar space savings at SAP customers:

- Around 50% space savings after applying row compression on an SAP ERP Unicode system (on SQL Server 2008 R2)
- Around 75% space savings when using page compression

The following compression ratios were measured using a 5 TB customer database of an SAP ERP Unicode system. Index fragmentation did not have any impact since the database was completely reorganized before the compression.

Compression Type	Percentage of original
No compression	100%
Row compression on data (clustered index) using SQL Server 2008	83.35%
Page compression on data (clustered index) using SQL Server 2008	46.36%
Row compression on data (clustered index) using SQL Server 2008 R2	64.28%
Row compression on data and index (clustered and non-clustered index) using SQL Server 2008 R2	54.75%
Page compression on data and index (clustered and non-clustered index) using SQL Server 2008 R2	24.88%

Configuring SQL Server Data Compression with SAP

Once you have applied the correction instructions of SAP note 1459005, the SAP Data Dictionary creates all new indexes with the same compression type as their table. New tables are created with compression type ROW. This default value might change in the future to compression type PAGE. You can overwrite the default compression type by adding one of the following lines in the SAP Default Profile:

- `dbs/mss/compression = page`
- `dbs/mss/compression = row`
- `dbs/mss/compression = none`
- `dbs/mss/compression = default`

Note: The profile parameter dbs/mss/compression has to be set in the SAP Default Profile. It does not work for the SAP instance profile to ensure the same setting for all SAP application servers.

The name of the profile parameter `dbs/mss/compression` must be in lower case. When setting the value `default`, the compression type for new tables is determined by the current version of the SAP Data Dictionary. When setting `page`, `row` or `none`, the compression type for new tables is determined by the profile parameter.

You can edit the default profile manually or use SAP transaction RZ10. However, the parameter `dbs/mss/compression` is not known in RZ10. Nevertheless, the parameter takes effect once you have activated it and restarted your SAP system.

Keep in mind that the compression type of already existing tables is not changed. The compression type of indexes always depends on the compression type of the table.

Compressing Existing Tables Using MSSCOMPRESS

SAP released the report MSSCOMPRESS in SAP note 1488135. You can use the report to compress existing tables and indexes using the data compression feature of SQL Server 2008 (and higher). You can choose between the compression types NONE, ROW, or PAGE for the data (heap or clustered index) and the (non-clustered) indexes. MSSCOMPRESS performs the compression in dialog or using an SAP batch job. MSSCOMPRESS is particularly useful for these scenarios:

- Compressing (non-clustered) indexes after activating the index compression support for SAP
- Re-compressing a SAP Unicode database to benefit from the improved Unicode compression of SQL Server 2008 R2
- Performing page compression

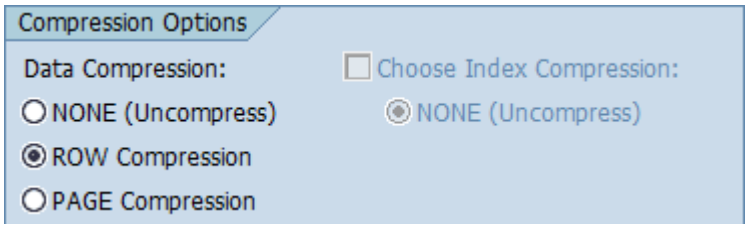
Prerequisites

If the report MSSCOMPRESS was not already imported into your SAP system by a basis support package, you have to import it manually (see below).

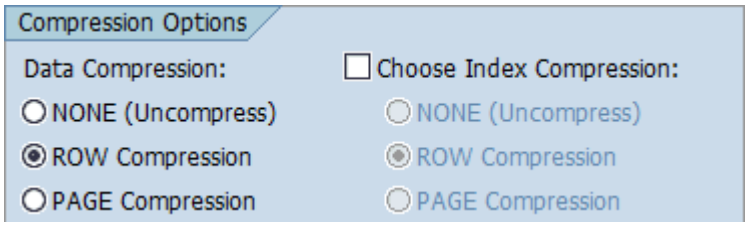
You can only use MSSCOMPRESS for SQL Server 2008 or higher. You need the same privileges to use MSSCOMPRESS as you need for the SAP database utility SE14 (authority object S_DEVELOP).

Note: The Authorization check of report MSSCOMPRESS was refined in March 2011. After applying the correction instruction of SAP note 1567962, you can run MSSCOMPRESS also with DBA administration authorization.

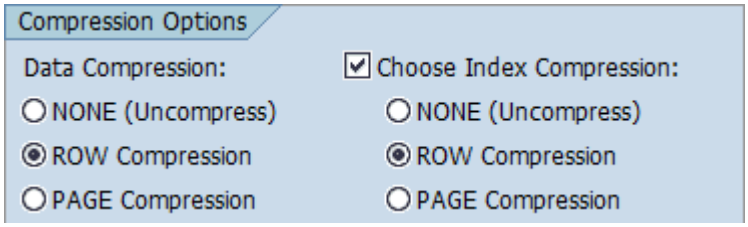
Before using MSSCOMPRESS, implement the correction instructions of SAP note 1459005. This activates the index compression support in the SAP data dictionary. As long as the SAP data dictionary is not up-to-date, you can only compress the data (heap or clustered index) using MSSCOMPRESS.



When you have applied SAP note 1459005, you can compress data and (non-clustered) indexes using the report MSSCOMPRESS.



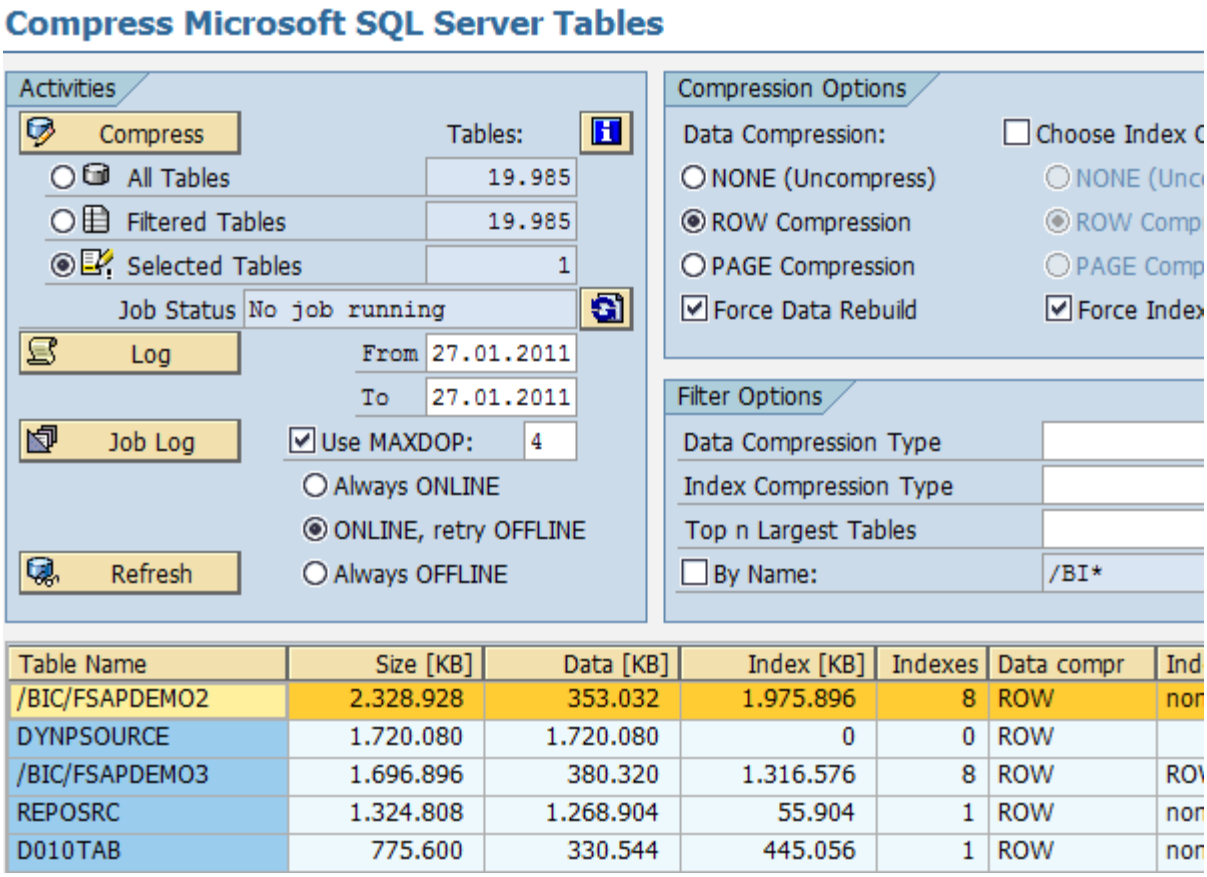
The default compression type for report MSSCOMPRESS is taken from the default compression type of the SAP data dictionary (changed from ROW to PAGE compression in May 2011). The index compression type is synchronized with the data compression type. However, you can manually choose the index compression type by checking the relevant checkbox:



Starting MSSCOMPRESS

Use SAP transaction SA38 or SE38 to start the ABAP report MSSCOMPRESS.

To get a brief description of MSSCOMPRESS, press the *i* button or choose *Application Help* from the *Help* menu. In the following, MSSCOMPRESS is described in detail.



In the lower section of the screen, a list of all database tables of your SAP ABAP system is displayed. The list does not include tables of an installed Java schema, since the data dictionary for Java does not support compression. There are SAP Java solutions, such as the SAP Enterprise Portal (EP), that have huge database tables. However, almost all data in an EP system is stored in fields of SQL Server data type IMAGE or VARBINARY(MAX). These data types cannot be compressed in SQL Server 2008 (R2).

The list of tables displayed in MSSCOMPRESS contains the number of indexes, the data/index size, and the data/index compression types for each table. The tables in the list are sorted by total size (data + index size). The number of indexes does not contain the clustered index. A clustered index (or primary key constraint) is considered to be DATA and all non-clustered indexes are INDEXES. The data compression type can be “ROW”, “PAGE” or “none”. For a partitioned table you might see a combined compression type, such as “ROW, PAGE”. This is the case, some partitions are row-compressed and some partitions of the same table are page-compressed. The same applies for the index compression type. When a table does not have any non-clustered index, the number of indexes is shown as 0 and the index compression type is empty (“ “).

Workflow Overview

To start the compression, do the following:

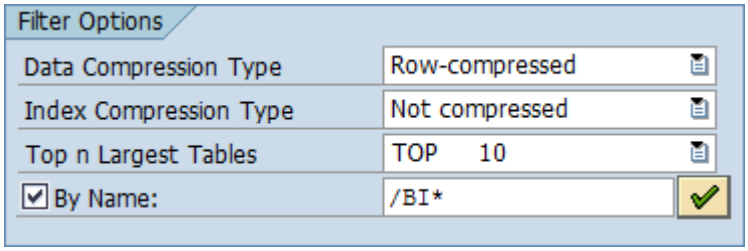
1. Choose the tables you want to compress (by filtering and selecting).
2. Choose the options (compression types and runtime options).
3. Choose the type of compression run (dialog or batch).

You can compress the SAP database in chunks. For example, you might choose the 10 largest non-compressed tables using the filters in MSSCOMPRESS and schedule the compression for the weekend. The other tables can then be compressed on following weekends.

Note: Compressing huge tables results in high resource consumption (CPU, I/O, blocking database locks). Therefore, do not run the compression during peak hours. The normal operation of your system might be affected by the compression jobs, even when using the *ONLINE* option. For more information about the *ONLINE* option, see SAP note 1006887.
Make sure that there is sufficient free space in the data and log files of the SAP- and tempdb-database. It is recommended to use the SQL Server recovery model *Bulk-logged* during compression to minimize the transaction log space needed. However, there are specific guidelines and recommendations when using SQL Server Database Mirroring. For more information, see SAP notes 965908 and 1550337.

Choosing the Tables

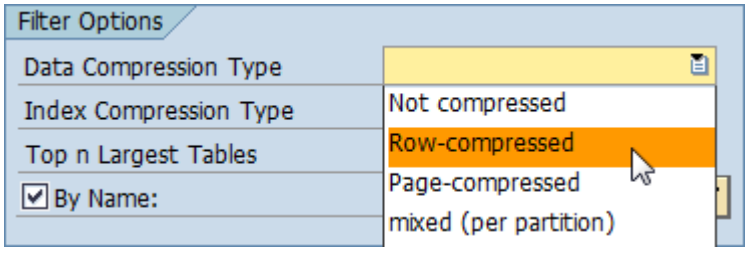
To choose the tables you want to compress, you can first filter the list of tables and then select the tables within the filtered list. Applying a filter modifies the displayed list of tables. You can apply multiple filters at the same time. In the following example, all four possible filter types are applied:



After applying these filters, the list contains the 10 largest tables, whose names start with “/BI”, data compression is “ROW”, and index compression is “none”. You can then further select the tables within the filtered list by clicking on them. Use the SHIFT and CTRL keys while clicking to expand your selection. In this example, three tables are selected:

Table Name	Size [KB]	Data [KB]	Index [KB]	Indexes	Data compr	Index compr
/BIC/FSAPDEMO2	2.328.928	353.032	1.975.896	8	ROW	none, ROW, P
/BIC/DMMCUBE7P	64	16	48	3	ROW	none
/BIC/DMMINVP	64	16	48	3	ROW	none
/BIC/DMMCUBE5P	64	16	48	3	ROW	none
/BIC/DMMCUBERTP	64	16	48	3	ROW	none
/BIC/DMMCUBERT1	48	16	32	2	ROW	none
/BIC/DMMCUBE51	48	16	32	2	ROW	none
/BIC/DMMINVT	48	16	32	2	ROW	none
/BIC/DMMCUBE71	48	16	32	2	ROW	none
/BIC/DMMCUBERTT	32	16	16	1	ROW	none

Choose the empty entry from the pop-up menu to remove a filter again. The name filter works in a slightly different way. You can enable and disable the name filter using the check box. By default, the filter criteria is “/BI*”, which filters all SAP BW tables. After changing the filter criteria, you have to press the ENTER key (or the ENTER button on the right-hand side of the filter criteria).



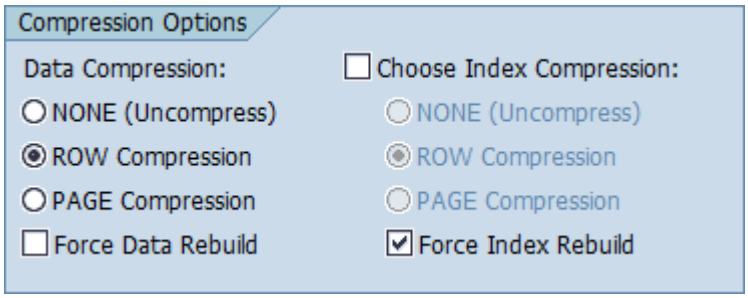
Choosing the Options

The compression options define the desired compression type of the tables after the compression run. Do not confuse them with the filter options *Data Compression Type* and *Index Compression Type*. They are used to choose the tables to be compressed. The filter options apply to the compression type of a table before the compression run.

You can choose the desired data and index compression type using radio buttons. The SQL statement used for compression is

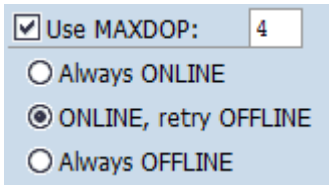
```
ALTER [TABLE| INDEX] … REBUILD WITH (DATA_COMPRESSION = [ROW | PAGE | NONE] …)
```

Before starting to compress, MSSCOMPRESS checks the current compression type for each table and index. If the current compression type fits the desired compression type, nothing is done for the index concerned. However, you can force compression of these tables and indexes independently of the current state by choosing the relevant checkboxes (*Force Data Rebuild* and *Force Index Rebuild*). These options are needed if you want to re-compress an already compressed table. This is particularly useful after upgrading SQL Server 2008 to SQL Server 2008 R2, to benefit from the compression improvements for Unicode strings.



MSSCOMPRESS automatically removes the vardecimal storage format of a table when compressing it. You can change this in the menu *Goto => Advanced Options*. However, we do not see any advantage in changing this default behavior.

The compression options have an impact on the result of the compression run. There are two options that have an impact on the compression run itself: MAXDOP and ONLINE.



You can configure the maximum number of threads used to execute a single SQL statement with the SQL Server configuration option max degree of parallelism (MAXDOP). For an SAP system, this is typically set to 1. To overwrite this configuration for the compression run, select the checkbox *MAXDOP* and enter a value greater than 1. MSSCOMPRESS compresses table-by-table and index-by-index. It does not compress multiple tables at the same point in time, even if *MAXDOP* is set. If you want to compress many tables in parallel, schedule a few compression runs of different tables for the same period. The benefit of parallel compression runs depends on your hardware resources.

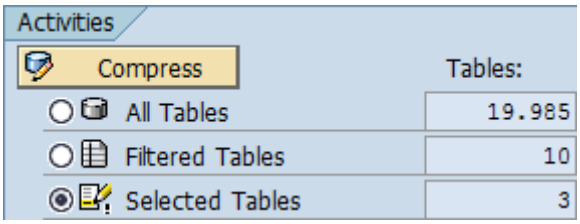
When choosing *Always ONLINE*, the compression does not acquire table locks for the duration of the compression, which results in less concurrency. However, the compression fails using this option for tables with a text or image field. These tables can be compressed either with the option *Always OFFLINE* or *ONLINE, retry OFFLINE*. The latter results in compressing all tables online if needed, followed by a second chance offline.

Note: There used to be an older version of MSSCOMPRESS, which had an ONLINE checkbox instead of the radio buttons. In this version, MSSCOMPRESS always performed an offline retry when the online compression failed.

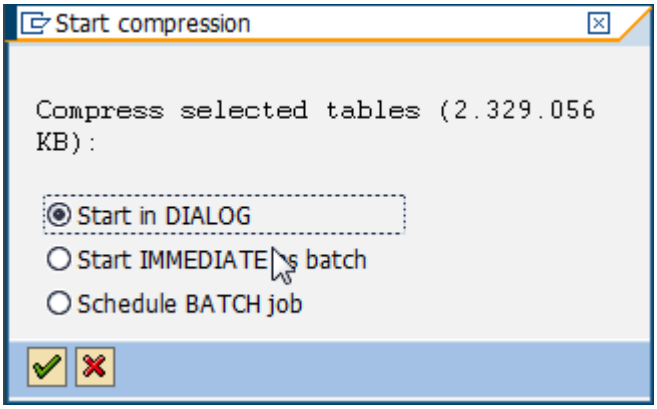
Generally speaking, the online option makes sense when compressing a few tables in dialog during normal working hours. It does not make sense when compressing many tables in a batch job during off-peak hours.

Starting the Compression

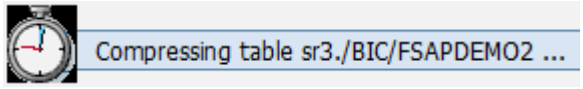
Use the radio buttons to choose what you want to compress. By default, all selected tables are compressed. You can also choose the filtered tables (all tables displayed in the list) or all tables (of the ABAP schema).



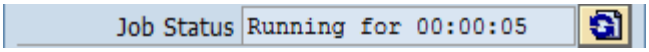
A dialog box appears after you press the *Compress* button. Here you can choose whether you want to start the compression run in dialog, start it immediately as a batch job, or schedule a batch job for off-peak hours.



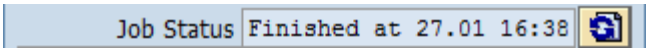
When compressing in dialog mode, a progress indicator is displayed at the bottom left corner of the window. When the compression run has finished, the compression type and data/index size is refreshed in the list of tables. Afterwards the current filter is applied again. The selected tables are still selected, which makes it easier to see the result of the compression run. However, this might not be the case when using filters, such as the filter option *Row-compressed*. After page-compressing a row-compressed table, the filtered list no longer contains the table (because the filter criteria no longer applies for this table).



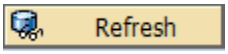
Once you have started a compression run as a batch job, you can see the job status in the main screen of MSSCOMPRESS.

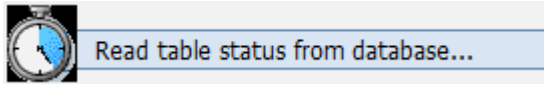


However, you have to update the job status manually by pressing the button on the right-hand side of the job status field.



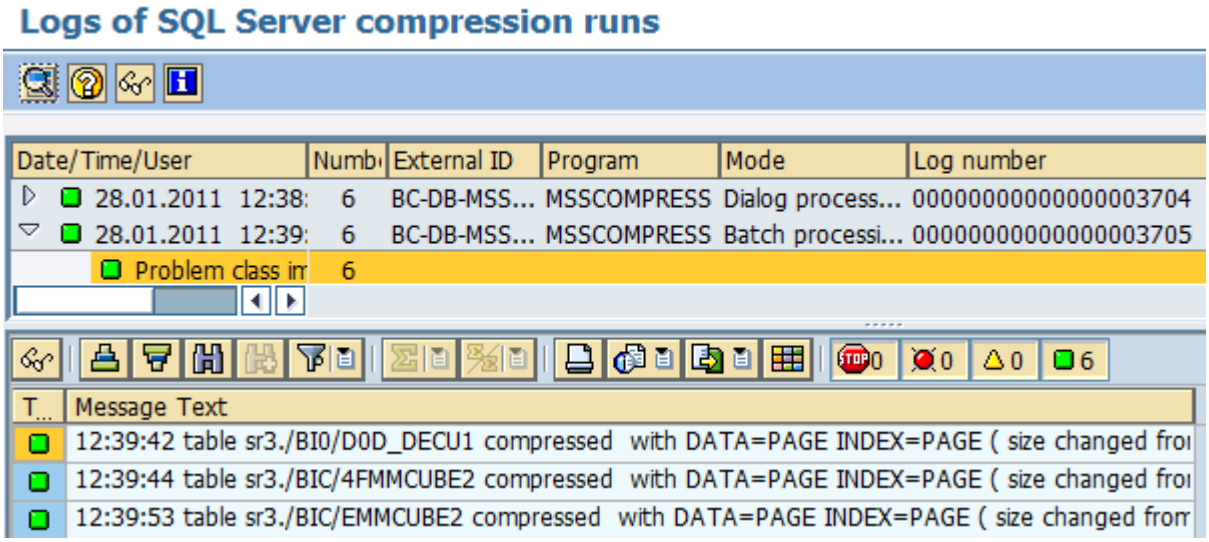
When using a batch job, compression types and data/index size in the list of filtered tables is not updated automatically. However, you can do this manually by pressing the *Refresh* button.



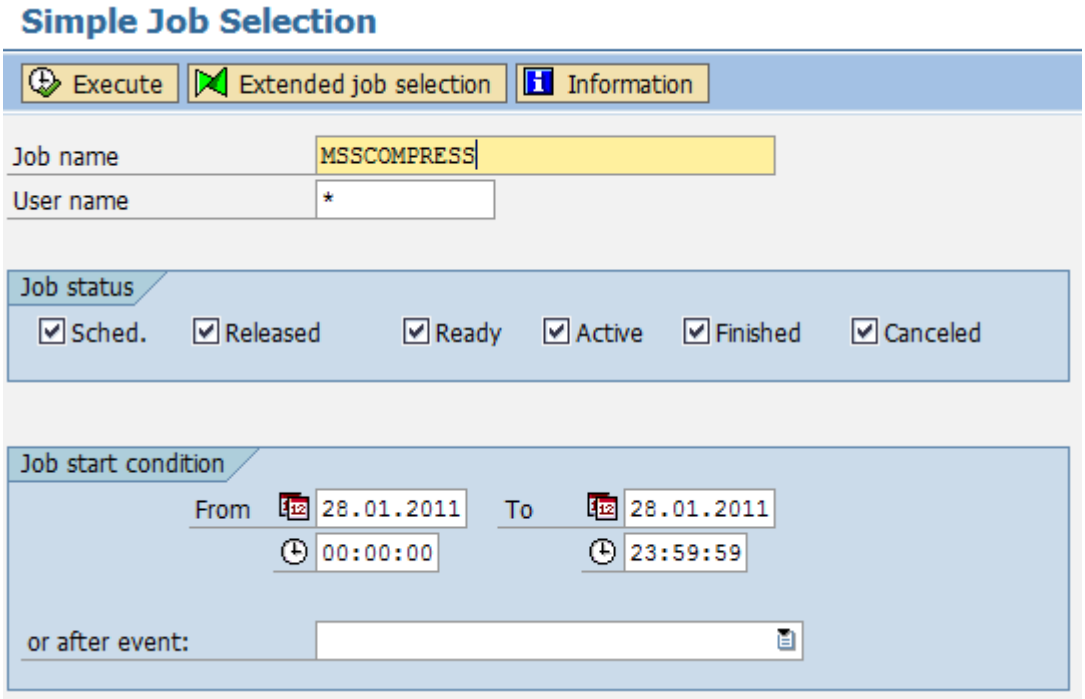


Checking the Log Files

MSSCOMPRESS creates a compression log for each table or index compression. The log is not stored as a file: instead, it uses the standard SAP Application Log. You can display the compression logs by pressing the *Log* button. By default, only the logs from the current day are displayed. However, you can change the *From* date and *To* date before starting the application log.



If the compression is running in an SAP batch job, a job log is also written. Press the *Job Log* button to see the standard job selection screen of SAP. Here you can check the status and job log of any SAP batch job.



You can read the job log while the job is still running. The job contains entries about the progress of the whole job ("… compressed 15% …"). Progress is calculated by the size of already compressed tables compared to the size of all tables that are scheduled for compression. To save space in the database, MSSCOMPRESS always starts the compression with the smallest tables of the compression run. Therefore, there seems to be only very little progress after the first tables have already been compressed.

Job Log Entries for MSSCOMPRESS / 12394000

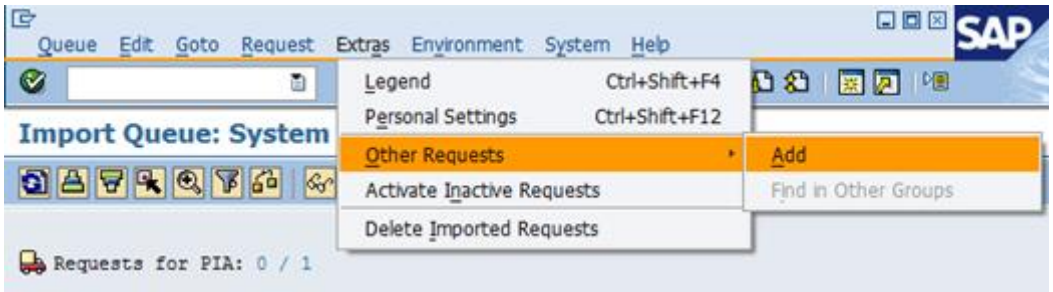
Long text | Previous Page | Next page

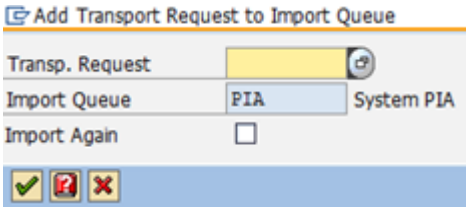
Job log overview for job: MSSCOMPRESS / 12394000

Date	Time	Message text
28.01.2011	12:39:40	Job started
28.01.2011	12:39:41	Step 001 started (program MSSCOMPRESS, variant &0000000000002, user ID C5000
28.01.2011	12:39:41	alter table [sr3].[/BI0/D0D_DECU1] rebuild with (data_compression = PAGE,
28.01.2011	12:39:42	alter index [/BI0/D0D_DECU1~010] on [sr3].[/BI0/D0D_DECU1] rebuild with (
28.01.2011	12:39:42	table sr3./BI0/D0D_DECU1 compressed with DATA=PAGE INDEX=PAGE (size changed
28.01.2011	12:39:42	... compressed 5 percent ...
28.01.2011	12:39:42	alter table [sr3].[/BIC/4FMMCUBE2] rebuild with (data_compression = PAGE,
28.01.2011	12:39:43	alter index [/BIC/4FMMCUBE2~020] on [sr3].[/BIC/4FMMCUBE2] rebuild with (
28.01.2011	12:39:43	alter index [/BIC/4FMMCUBE2~040] on [sr3].[/BIC/4FMMCUBE2] rebuild with (
28.01.2011	12:39:43	alter index [/BIC/4FMMCUBE2~050] on [sr3].[/BIC/4FMMCUBE2] rebuild with (
28.01.2011	12:39:44	alter index [/BIC/4FMMCUBE2~P] on [sr3].[/BIC/4FMMCUBE2] rebuild with (da
28.01.2011	12:39:44	table sr3./BIC/4FMMCUBE2 compressed with DATA=PAGE INDEX=PAGE (size changed
28.01.2011	12:39:44	... compressed 12 percent ...
28.01.2011	12:39:51	alter table [sr3].[/BIC/EMMCUBE2] rebuild with (data_compression = PAGE,
28.01.2011	12:39:52	alter index [/BIC/EMMCUBE2~010] on [sr3].[/BIC/EMMCUBE2] rebuild with (da
28.01.2011	12:39:52	alter index [/BIC/EMMCUBE2~020] on [sr3].[/BIC/EMMCUBE2] rebuild with (da
28.01.2011	12:39:53	alter index [/BIC/EMMCUBE2~040] on [sr3].[/BIC/EMMCUBE2] rebuild with (da
28.01.2011	12:39:53	alter index [/BIC/EMMCUBE2~050] on [sr3].[/BIC/EMMCUBE2] rebuild with (da
28.01.2011	12:39:53	alter index [/BIC/EMMCUBE2~P] on [sr3].[/BIC/EMMCUBE2] rebuild with (data
28.01.2011	12:39:53	table sr3./BIC/EMMCUBE2 compressed with DATA=PAGE INDEX=PAGE (size changed
28.01.2011	12:39:53	... compressed 25 percent ...
28.01.2011	12:39:54	alter table [sr3].[/BIC/FMMCUBE2] rebuild with (data compression = PAGE,

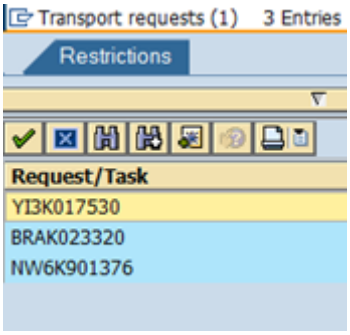
Importing the Transport of MSSCOMPRESS

If the report MSSCOMPRESS was not already imported into your SAP system by a basis support package, you have to import it manually. The SAP basis administrator should do this. You first have to unzip the cofile (`Kxxx.SID`) and data file (`Rxxx.SID`) of the transport, which are attached to SAP note 1488135. Then copy the files to the cofiles (`usr\sap\trans\cofiles\`) and data directory (`usr\sap\trans\data\`). Then add the request to the import queue in SAP transaction STMS.



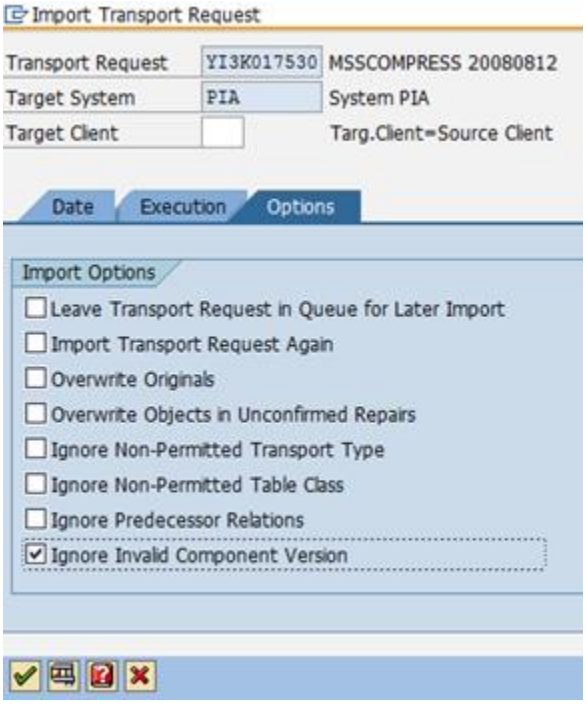


For field *Transport Request*, use the F4 help to choose your transport.



Select the Request (`SIDKxxx`) and press the button *Import Request*  .

For SAP NetWeaver releases 7.2 and higher, you might have to choose the import option *Ignore Invalid Component Version*.

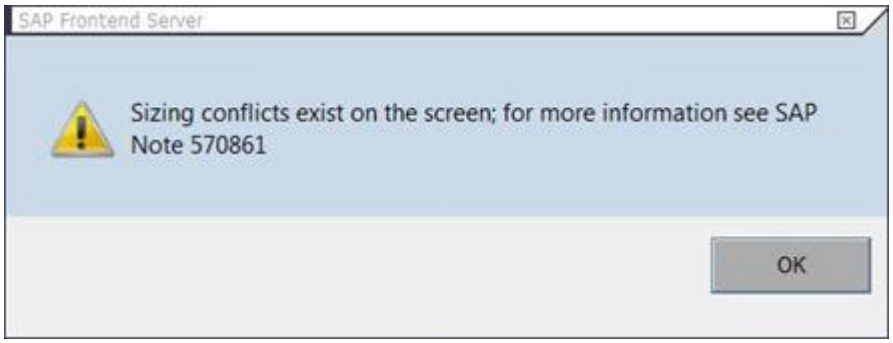


This procedure might vary slightly, depending on the SAP release. For more information, contact your SAP basis administrator.

Known Issues

Sizing Conflicts

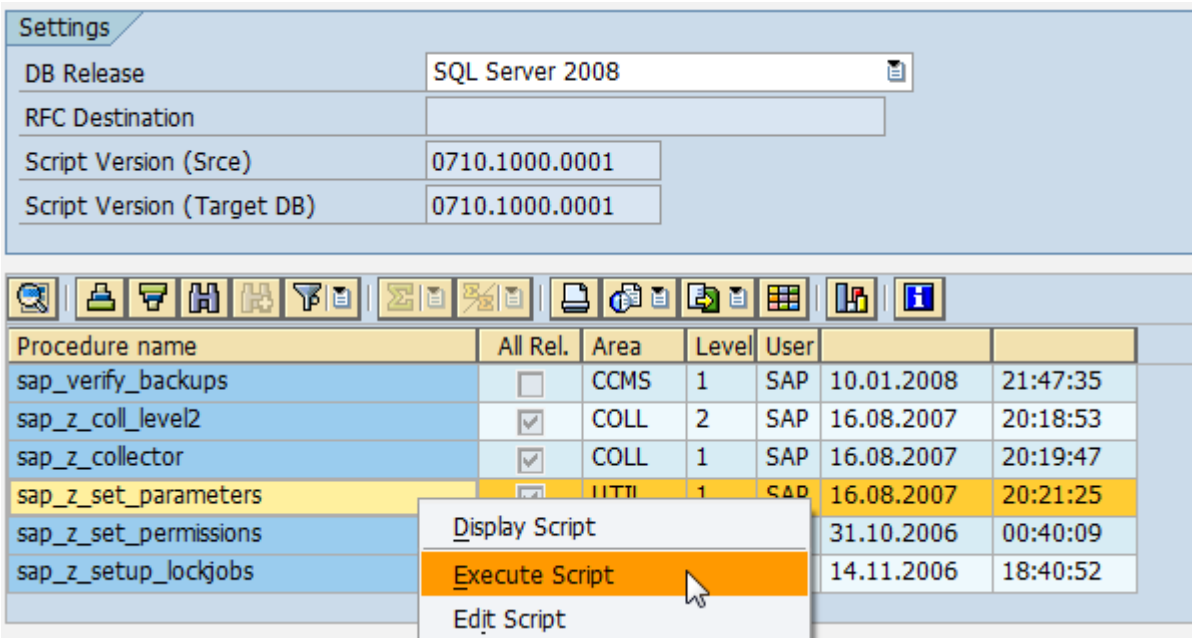
After pressing the *Compress* button, you might see the following warning:



You can ignore this warning by pressing *OK* because it has no impact at all. The root cause of this warning is nothing to do with report MSSCOMPRESS. It is a known bug in the SAP kernel, which is fixed by SAP note 1309615. After applying the newest SAP kernel patch, the warning no longer appears.

Autostats Option of VBDATA

SAP turns off the automatic update statistics for a few tables such as VBDATA. In addition, VBDATA has a field of type IMAGE or VARBINARY(MAX). Compressing VBDATA with the online option fails because of the IMAGE field. When repeating the compression offline, the autostats option of VBDATA changes erroneously. This issue is due to be fixed in future SQL Server versions.
Since MSSCOMPRESS can retry the compression offline, you might run into this issue. However, the changed autostats option does not have any impact on most customer systems. Anyway, you can easily recover the original SAP configuration settings by running the SQL script `sap_z_set_parameters` using report MSSPROCS:



SQL Error 402

A handful of customers ran into SQL error 402 when running MSSCOMPRESS. This is caused by a faulty configuration of the SAP system. For some reason they had set an undocumented profile parameter. If you run into SQL error 402 in report MSSCOMPRESS, contact SAP support.

Changing Compression Type for SAP Installation and Upgrade

SAP uses the executable `r3load` for creating new tables during the installation, migration and upgrade. R3load does not take care of SAP profile parameters. Since there is no way to keep the compression type of the SAP Data Dictionary synchronized with the compression type of `r3load`, the compression type of new (and even existing) tables might change during an SAP upgrade. Therefore, re-compress these tables using report MSSCOMPRESS after an SAP upgrade.

You can change the default compression type of `r3load` by modifying the file `DDLMSS.TPL`, which is located in the SAPINST directory and on the exports DVDs. The template file `DLMSS.TPL` starts with the following lines:

```
prikey: BEFORE_LOAD ORDER_BY_PKEY
seckey: AFTER_LOAD

cretab: CREATE TABLE &tab_name&
( /{ &fld_name& &fld_desc& /-, /} )

# to deactivate row compression on SQL 2008 and higher
# add &norowcompression& after the closing bracket:
# cretab: CREATE TABLE &tab_name&
# ( /{ &fld_name& &fld_desc& /-, /} ) &norowcompression&
```

In February 2008, SAP released a patched `r3load`, as described in SAP note 1143246. This version is contained in all SAP/SQL installation und upgrade DVDs before September 2010. It creates tables with compression type ROW and indexes with compression type NONE when connected to SQL Server 2008 or higher. You can disable the row compression for tables by adding the hint "&norowcompression&" in the *cretab* section of `DDLMSS.TPL`.

In September 2010, a new `r3load` was released, as described in SAP note 1505884. It still creates tables with compression type ROW and indexes with compression type NONE. However, you can override the compression type used by `r3load` for tables and indexes by adding hints in the *cretab* and *creind* section of `DDLMSS.TPL`. The following hints are possible:

- **&norowcompression&:** r3load does not use any compression type
- **&compression_none&:** r3load uses compression type NONE
- **&compression_row&:** r3load uses compression type ROW
- **&compression_page&:** r3load uses compression type PAGE

The template file `DDLMSS.TPL` is created using the executable `r3ldctl.exe`. The newest version of `r3ldctl` creates a template with page compression for tables and indexes.

Configure Page Compression for System Copies and Migrations

Always read SAP note 888210 before starting a homogeneous or heterogeneous system copy. To benefit from page compression, you have to perform the following steps:

1. Apply SAP note 1581700 on the source system.
   The correction instruction in this SAP note updates the default compression type of the SAP data dictionary to page compression. This will affect the target system, too.
2. Use the newest SAP kernel including `r3load` and `r3ldctl` for export and import.
   The newest `r3ldctl` creates a correct `DDLMSS.TPL` file during the export on the source system. `R3load` then creates all tables page-compressed during the import into the target system.
3. Remove the profile parameter `dbs/mss/compression` on the target system.
   Since the default in the SAP data dictionary is already page compression, there is no need any more to override the default.

Copyright