SAP® MaxDB™ Expert Session

SAP® MaxDB™: Introduction into DBM Server

Heike Gursch Sept. 17, 2013

Public

SAP® MaxDB™ – Expert Session

Introduction into SAP ® MaxDB™ DBM Server

Heike Gursch
Oksana Alekseious
IMS MaxDB/liveCache Development Support
Sept 17, 2013

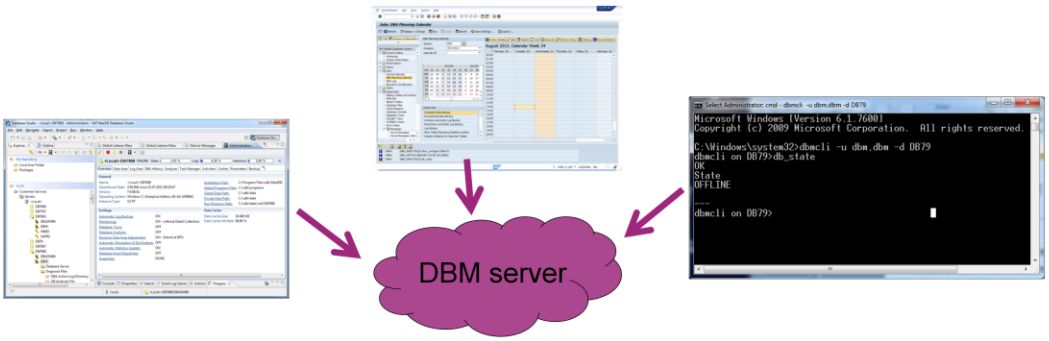
Agenda

- Tasks and Concept of the DBM Server
- What is the DBM Server used for? Choice of dbmcli commands
- Communication via Shared Memory
- Files for managing the DBM Server
- Problem analysis and ShM* (demo)
- Commands dbm_stop and db_drop
- Background DBM Server (demo)
- Scheduler
- User Event Dispatcher

Tasks of the DBM server

- process administrative commands
- establishes the connection from the database clients to the database kernel
- only administrative commands
- usually SQL commands do NOT communicate via the DBM server

- Clients: Database Manager CLI (dbmcli), Database Studio, SAP tools (via DBMRFC) like DBACockpit or DBA planning calendar (db13)



The DBM server establishes the connection from the database clients to the database kernel.

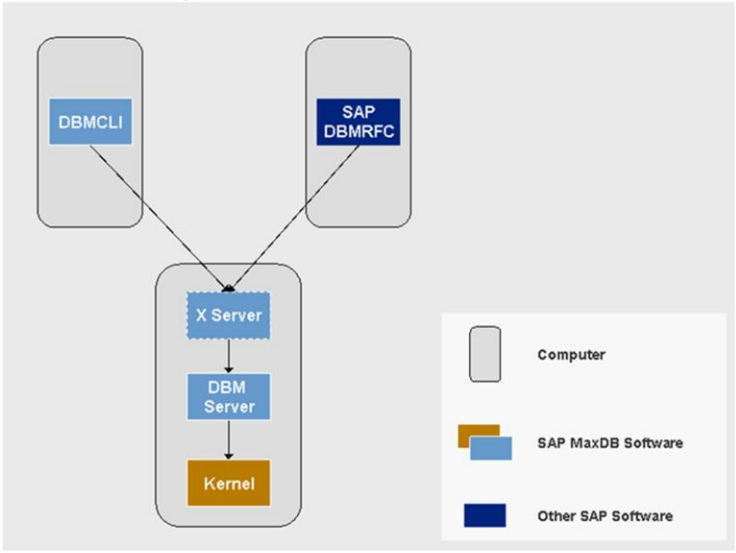
As a prerequisite you have to be logged on to the database as a database system administrator or DBM operator.

If administrative commands are sent remotely to the database host, a connection is first established via the remote SQL server which in turn starts a DBM server.

Usually SQL statements do not communicate via the DBM server, in case of a remote connection they use the x_server.

If an SQL statement is sent via sql_execute or db_execute then it will also communicate via a DBM server session.

Concept (Remote Access)



DBM Server

The DBM server is the server via which administration commands are executed. The database tools Database Studio, Database Manager, and Event Dispatcher use the DBM server. The DBM server always resides on the same computer as the database.

Database Manager

Database Manager consists of a client part and a server part (DBM Server).

The client part of Database Manager is available in the following variants:

Database Studio (see above)

Database Manager CLI (DBMCLI): Command line interface

Database Manager RFC (DBMRFC): Only in SAP systems, interface to the SAP system

(Another mechanism without dbmrfc is used as of Netweaver 7.30, 7.03 and 8.0 in combination with MaxDB 7.9.04 or higher.)

If the client part of Database Manager is located on another computer than the database, it connects to the database on the remote computer via an X Server.

Database Manager CLI is the client program of Database Manager. With this program you connect to the DBM server on the computer on which you want to create a new database or manage an existing database.

The Database Manager CLI database tool is used to administer database. The tool allows you to create, monitor, back up, and restore databases.

Database Manager CLI is a command line tool and is suited to both interactive operation and batch operation. It can be used on all operating systems supported by the database system.

General

- dbmsrv (UNIX) or dbmsrv.exe (Windows)
- located in <InstallationPath>/pgm
- Usually the DBM server is not explicitly started by the user but implicitly when an administrative DBM command is called via a client

Under UNIX the processes can be identified in the process list as dbmsrv (use: ps –ef | grep dbmsrv).

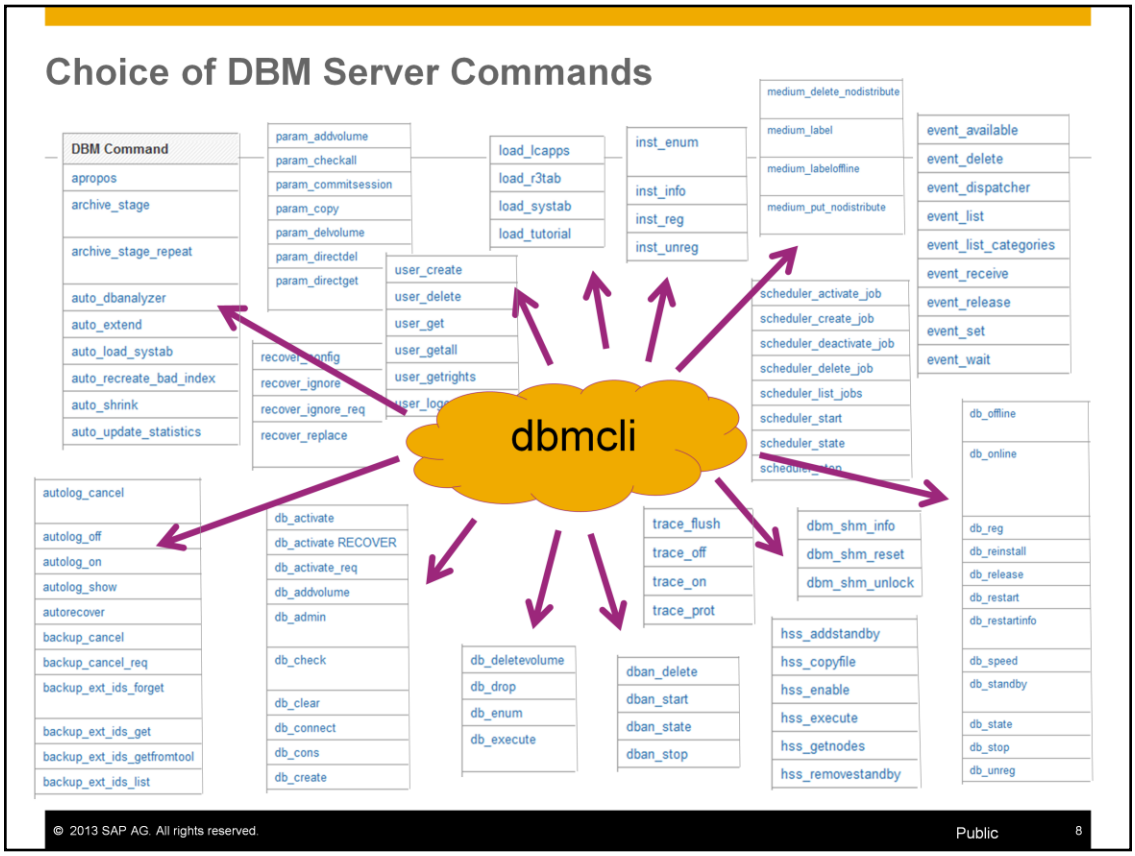
Under Windows you will find it as dbmsrv.exe.

The installation path can be identified by using the command

xinstinfo <DB name>

What is the DBM Server used for?

- Software Installation
- Creating and Deleting Databases
- Configuring Databases
- Controlling Databases
- Backing Up and Restoring Databases
- Managing Database Manager Operators
- Analyzing Databases
- Accessing Databases Using SQL
- Optimizing the SQL Access to a Database
- Using Events of the Database
- Using the Scheduler
- Managing Hot Standby Systems
- Managing the Database Manager Tool



This picture is not complete but gives an impression how powerful the functionality of the DBM server is.

If you are not sure about the exact names and options of a command the following commands help to identify the command names and how they are used.

Helpful DBM Commands:

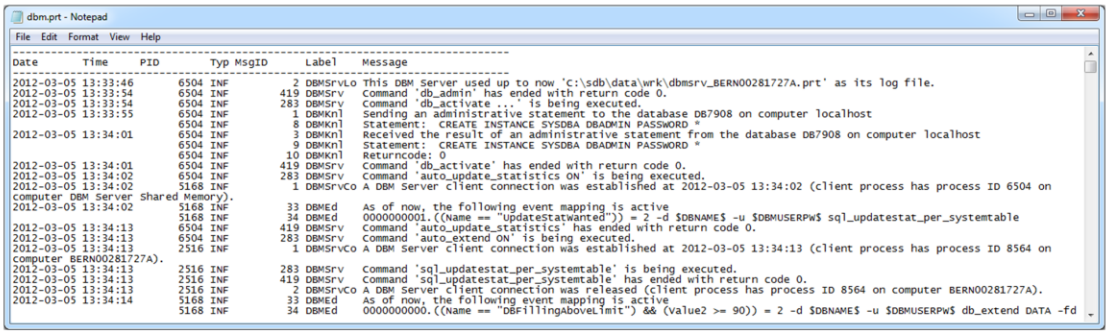
apropos - Searching for a DBM command

explain - displaying the help text for a DBM command

help - displaying a list of DBM commands

dbm.prt

- written by DBM server
- plaintext file in the RUNDIRECTORY
- filekey: DBMPRT
- old content overwritten cyclically
- the size can be enlarged with command dbm_protfilesize <size> [<size_unit>]
- contains administrative statements (SAVE statements included)
- some statements are truncated because of security reasons



The dbm.prt is the main logfile which is written by the DBM server. You will find all actions taken by the DBM server within this file. The file has a restricted size and if there are a lot of messages parts of the content will get lost as the file is overwritten cyclically.

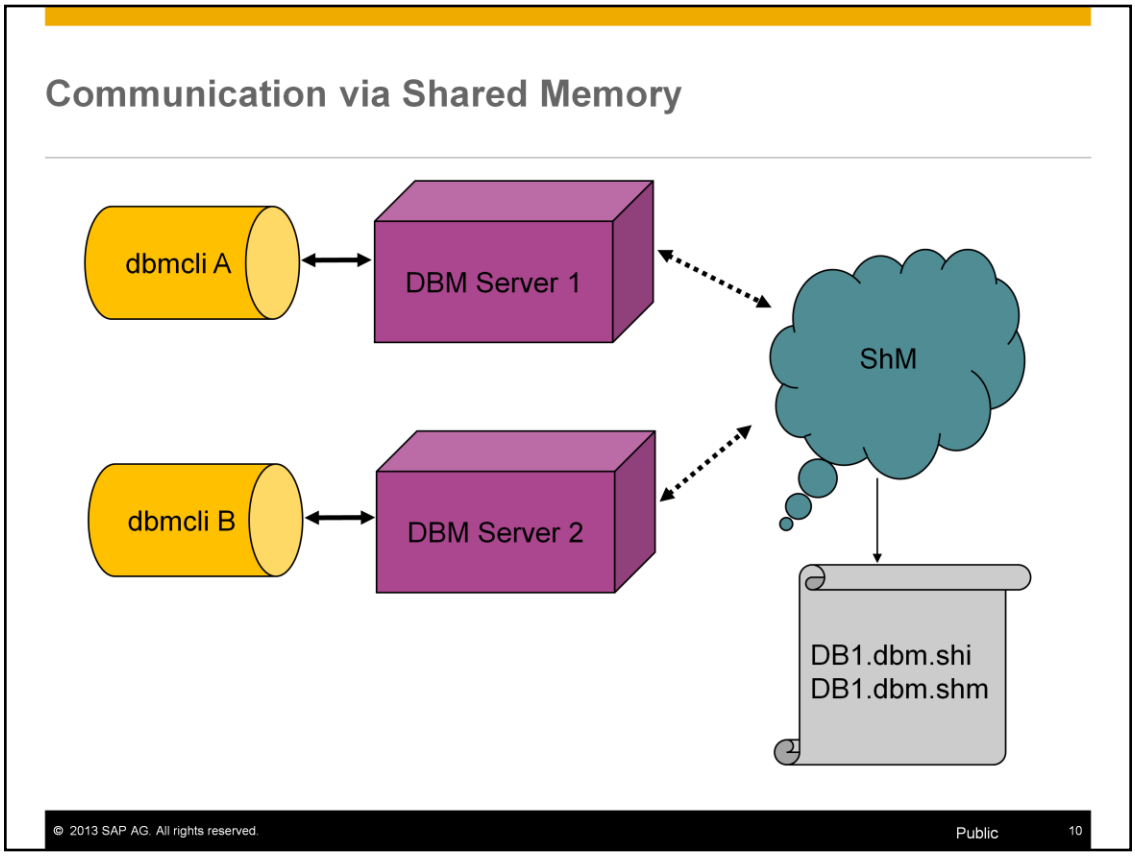
SAVE statements (== backup creation)

This log has a default size of maximum 1 GB and is overwritten cyclically. You can use the following DBM command to change the

size of the file:

dbm_protfilesize <size> [<size_unit>]

If no unit of measure (<size_unit>) is specified, the specification is in MB. You can use this DBM command to configure a maximum size of 2 GB. Change the size of the log dbm.prt only if requested to do so by SAP Support.



The synchronization of the different DBM servers of a database takes place using the shared memory segments <DB_name>.dbm.shm and <DB_name>.dbm.shi.

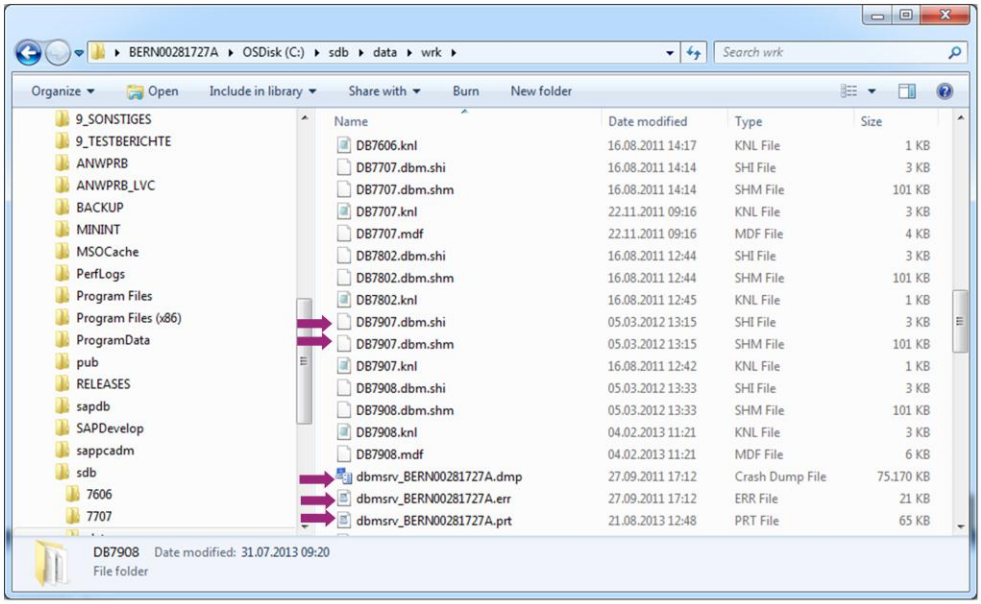
Furthermore the communication between the DBM servers is done via the shared memory segments. They are implemented in the system as a memory-mapped file (mmap). Make sure that the path for the shared memory is located on a local filesystem. If this file is located on a Network Share (NFS), deadlock situations might occur.

You can use the DBM command dbm_version to determine the directory in which the shared memory segments (<database_name>.dbm.shm and <database_name>.dbm.shi) are stored.

The entry SDBDBM_IPCLOCATION shows the directory:

- Usually /var/lib/sdb/dbm/ipc under UNIX
- Default directory for Windows:<GlobalDataPath>/wrk

Files for managing the DBM Server



<DB_name>.dbm.shi
<DB_name>.dbm.shm
dbmsrv_<server name>.dmp
dbmsrv_<server name>.err

Example for service database:

.M790802.dbm.shi
.M790802.dbm.shm

Files for managing the DBM Server – SHM and SHI

<dbname>.dbm.shi

- located in <sdbdbm_iplocation dir>
  (Default /var/lib/sdb/dbm/ipc for UNIX, <GlobalDataPath>/wrk for Windows)
- platform and version information (ASCII text at start of file)
- PIDs of all DBM Servers that are attached to this shared memory
- lock administration of ShM
- layout administration of ShM

<dbname>.dbm.shm

- located in <sdbdbm_iplocation dir>
- usage data (f.e. Resource locks, current commands, information about eventing)

ShM lets DBM Servers cooperate PER DATABASE only!

- DBM Server commands that are not addressed to a special database (inst_reg, db_enum,…)
  do not use it
- No communication between DBM Servers of different databases

Use the DBM command dbm_version to determine the directory in which the shared memory segments (<database_name>.dbm.shm and

<database_name>.dbm.shi) that the DBM server uses for the synchronization are stored.

The entry SDBDBM_IPCLOCATION shows the directory (for example, /var/lib/sdb/dbm/ipc for UNIX, /sapdb/data/wrk).

The GlobalDataPath can additionally be identified with the tool xinstinfo.

Or with the command:

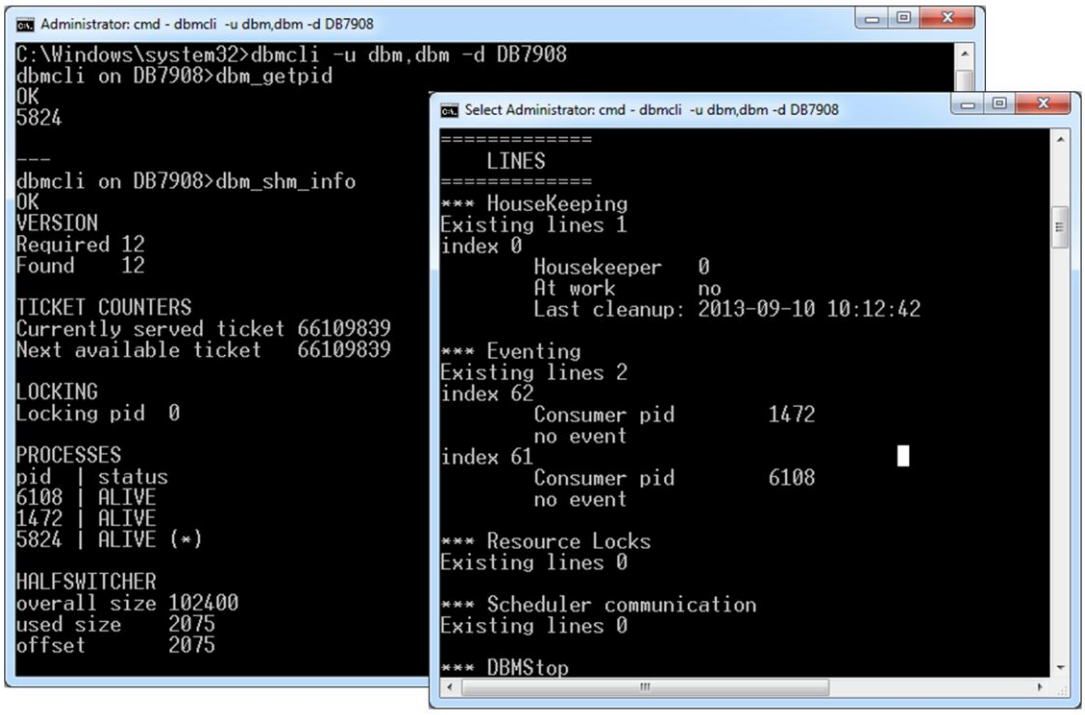
dbmcli –u <dbm>,<pw> -d <DB name> dbm_getpath

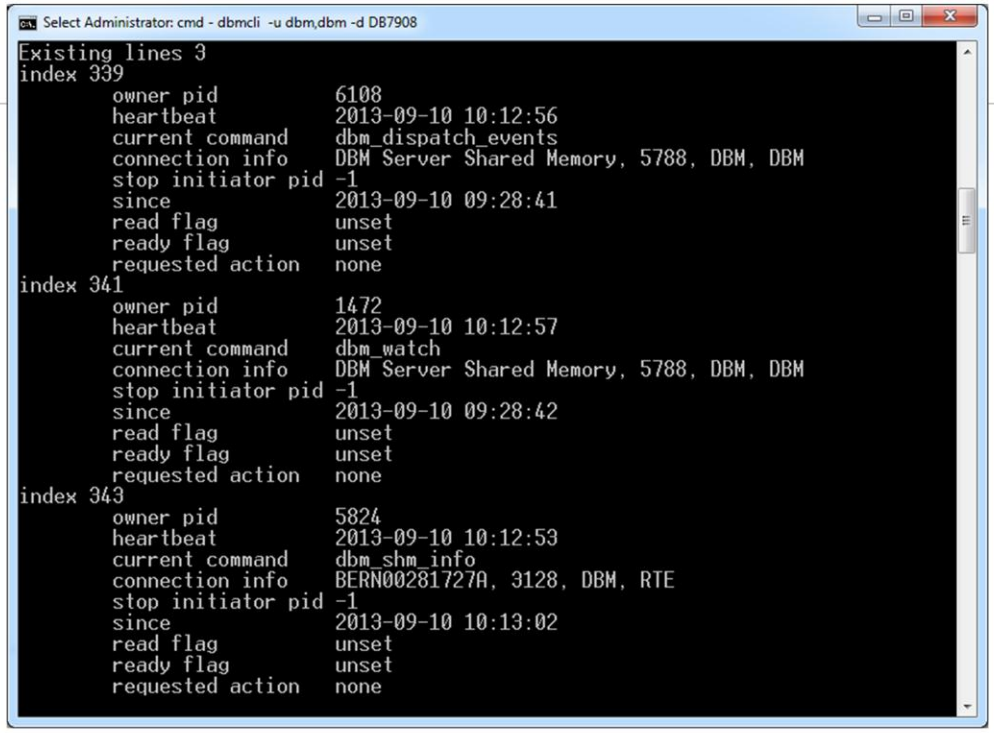
Command dbm_shm_info

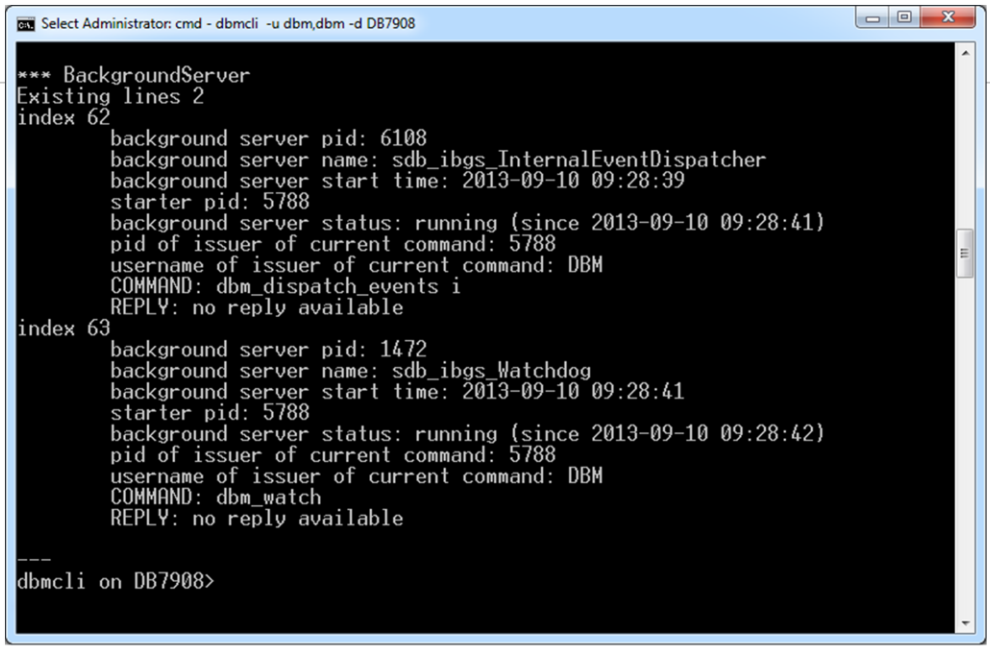
- Displays content of ShM
- Collects information from both files shi and shm
- Allows for checking DBM Server activity in a specific database

Identification of a DBM Server:
Use DBM server command **`dbm_getpid`**.

We will have a look into a ShM later during a demo …







Problem analysis and ShM

Problem:
If the DBM Server Shared Memory is in an inconsistent state, DBM Servers cannot start. No problem analysis is possible.

Possible solution:

```
<InstallationPath>/pgm/dbmshm [UNLOCK|DELETE|CHECK]
<sdbdbm_iplocation dir> <dbname>
```

- Can only be used on ShM of databases of the same release.
- To be used for maintenance of ShM.
- Is roughly the DBM Server command dbm_shm_info in a separate program

Examples for Error Situations (I)

ERR -24700 DBMSrv ERR_DBMSRV_NOSTART: Could not start DBM server.

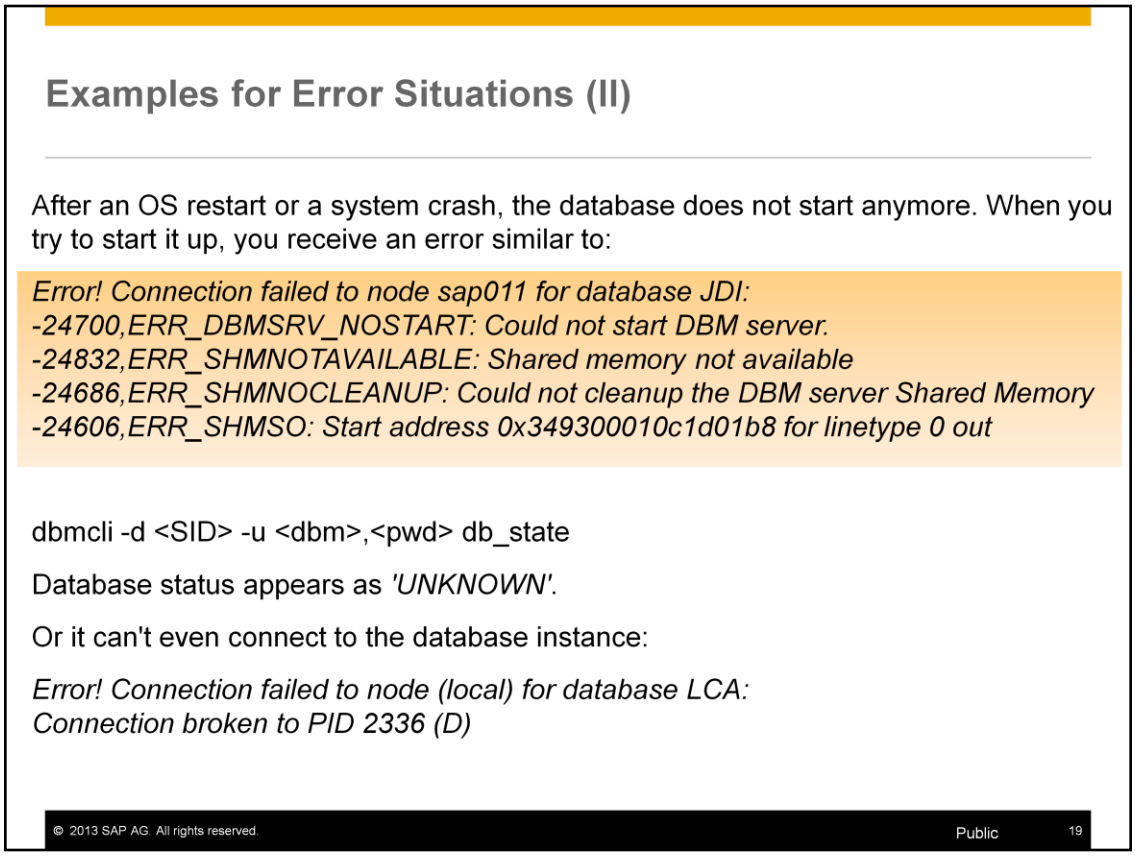
ERR -24832 DBMSrv ERR_SHMNOTAVAILABLE: Shared memory not available

ERR -24993 DBMSrv ERR_RTEEXT: Runtime environment error

ERR -24778 DBMSrv 1,could not create dir:/var/lib/sdb/dbm/ipc

ERR -24778 DBMSrv 13,No system errortext for ERRNO 13

The SDB user (usually sdb:sdba) must always be able to read and write in the target directory (usually /var/lib/sdb). No DBM server can run if this is not the case. They then terminate with the above error message.



Most likely this issue happens when the system is abnormally shutdown/restarted or after a crash (due to power outage, for example) while the database is running.

If the database is in ONLINE state and a OS restart is not properly done, the MaxDB cannot perform a 'clean' shutdown so the shared memory files of the DBM server reside in an inconsistent state, causing the old runtime to be present at the new database startup.

At OS level (process list), check whether any MaxDB/LiveCache processes are running. If any process is found, terminate it.

Locate the runtime files in the filesystem. These files are named as ***<DB name>*.dbm.shi and <DB name>*.dbm.shm,*** and are usually located under ***/sapdb/data/wrk*** *folders* (Windows***)*** and ***'/var/lib/sdb/dbm/ipc*** (UNIX).

Rename the files. I.e: ***LCA.dbm.shi_old***, ***LCA.dbm.shm_old***.

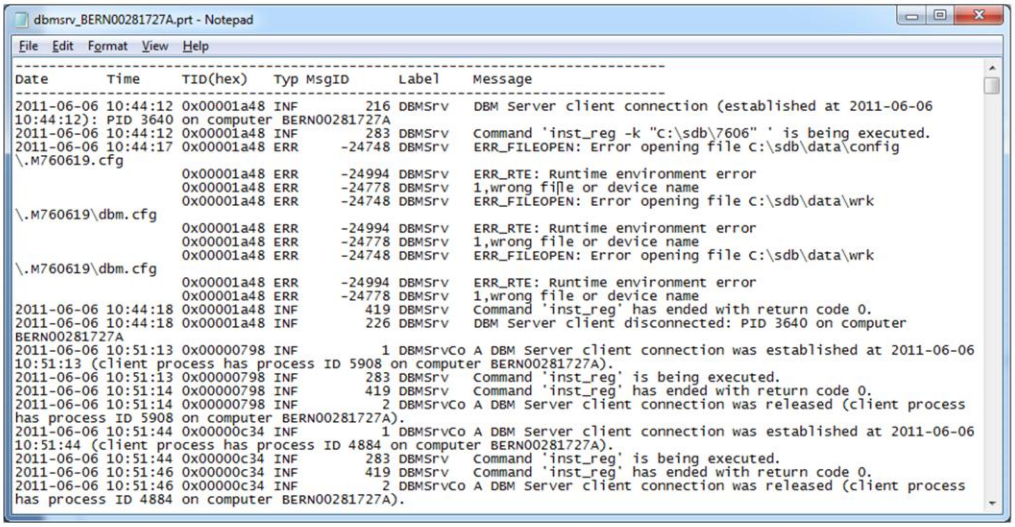
If the shared memory is not available any longer it is newly generated.

Another action might be necessary for cleanup but this is not related to DBM server:

Rename the pipe files (if existent) ***<SID>*** to ***<SID>.old*** in folders ***/sapdb/data/ppid, /sapdb/data/fifo*** and file ***db:<sid>*** to ***db<sid>.old*** in folder ***/sapdb/data/ipc.***

Files for managing the DBM Server – dbmsrv_<host>.prt

- Info messages (INF), warnings (WRN) and errors (ERR) concerning DBM server operation
- Contains information that is independent of a special database instance and start messages



Information that is independent of a special database or liveCache instance is written into dbmsrv_<host>.prt. F.e. commands like inst_reg will appear in this logfile.

Additionally the DBM server protocols its messages here in the start situation as long it is not related to a special instance. You will find a line indicating the position when it skips to the dbm.prt (and vice versa).

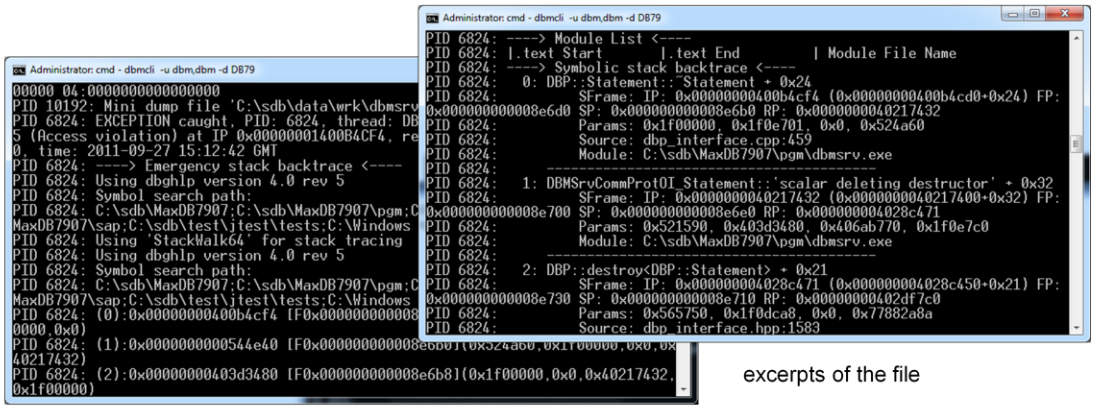
The file is located in the directory <GlobalDataPath>/wrk.

GlobalDataPath can be identified with the command:
dbmcli –u <dbm>,<pw> -d <DB name> dbm_getpath

The file is cyclically overwritten.

Files for managing the DBM Server – dbmsrv_<host>.err

- callstack of crashed DBM Server
- located in <GlobalDataPath>/wrk
- opened as soon as possible by each DBM Server
  (immediately after <GlobalDataPath>/wrk path is calculated)
- exists almost always (with a size of 0 bytes)



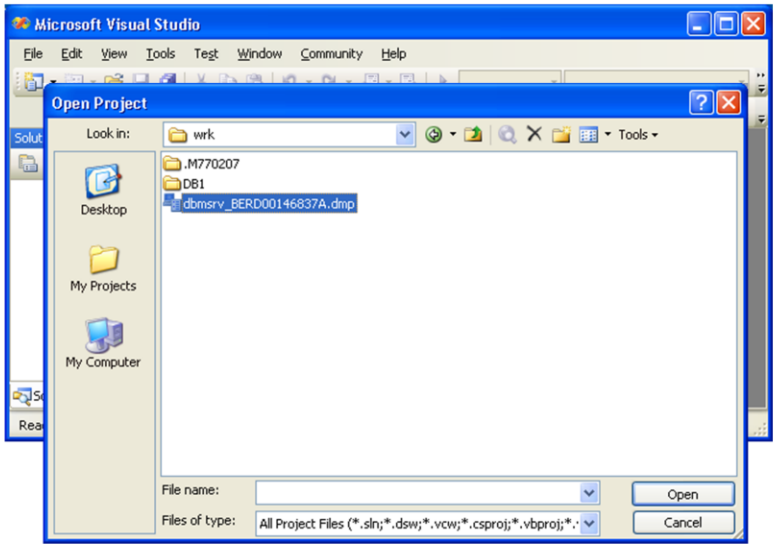
excerpts of the file

GlobalDataPath can be identified with the command:
dbmcli –u <dbm>,<pw> -d <DB name> dbm_getpath

Files for managing the DBM Server - dbmsrv_<host>.dmp

Crash Dump File for post-mortem analysis by developer

- entire process space of crashed DBM Server
- located in <GlobalDataPath>/wrk
- Analysis with Visual Studio
- Windows only



Under Windows there is an additional way that can be used by developers to identify errors from crash situations. If the backtrace from dbmsrv_<host>.err does not deliver enough information the Visual Studio can be used for further post mortem analysis. A debugger can be used and information about the values of variables can be identified.

dbm.cfg

- Database Manager configuration file for the database
- Located in the rundirectory of the database
- Changes on dbm.cfg (performed by Database Manager automatically) are written to log file dbmcfg.his.
- Examples for entries in dbm.cfg:
  DBADTL
  RunEventDispatcher
  AUTOSAVE

dbmsrv_diag

- used for enhanced error analysis to identify DBM server problems
- as of 7.9.08.08 and 7.8.02.32 part of software delivery
- program dbmsrv will be replaced by dbmsrv_diag
- traces dbmsrv_<pid>.trace written into rundirectory
- Tracing is activated if the following line is displayed in the output of command dbm_version: TRACING = YES
- After analysis make sure to reactivate the original DBM server

Commands dbm_stop and db_drop

`dbm_stop [-f] [PID]`

- Stops other DBM Servers that are related to the same database. Target DBM Servers are „asked“ to terminate themselves.

`dbm_stop` (no parameters)

- DBM Servers, that are idle don’t start executing their respective commands.
- DBM Servers, that are currently executing a command are allowed 30 seconds. New DBM Servers cannot be started during this period of time.
- If one of the DBM Servers does not respond to the stop request in time, dbm_stop fails and all DBM Servers are left alive.
- Terminated DBM Servers do not touch ShM anymore, they cannot execute commands. They stay alive for at most 5 minutes. If their respective DBM client tries to execute a command, they respond with an error message.

`dbm_stop <somepid>`

- Almost the same as above but only the DBM Server with PID <somepid> is targeted, other DBM Servers may start.

Reason for the invention of this feature:

- databases may only be dropped if no other DBM Server exists
- upgrade

Commands dbm_stop and db_drop

`dbm_stop -f <somepid>`

After at most 10 seconds the target DBM Server(s) are terminated (not only not touching ShM anymore, but really dead).

`db_drop [-f]`

performs implicit dbm_stop [-f] before actually dropping the database

This is implemented using multiple threads in each DBM Server process and another hitherto unmentioned file:

The DBM Server, that executes the dbm_stop (resp. db_drop) command, creates a file `<dbname>.dbm.exit` in <GlobalDataPath>/wrk. The existence of this file prevents new DBM Servers to be started. It is removed after the dbm_stop command has terminated (resp. the database is dropped).

DBM Servers are NEVER killed by means of the operating system! Therefore no foreign processes can be killed by accident.

This functionality is implemented with an extra checker. This is a thread that does nothing but checking stop requests in shared memory every 10 seconds.

Background DBM Server

Problem:
Long-running DBM Server commands like e.g. backup and restore block a DBM Server session and are bound to it irreversably.

Solution:
Background DBM Server

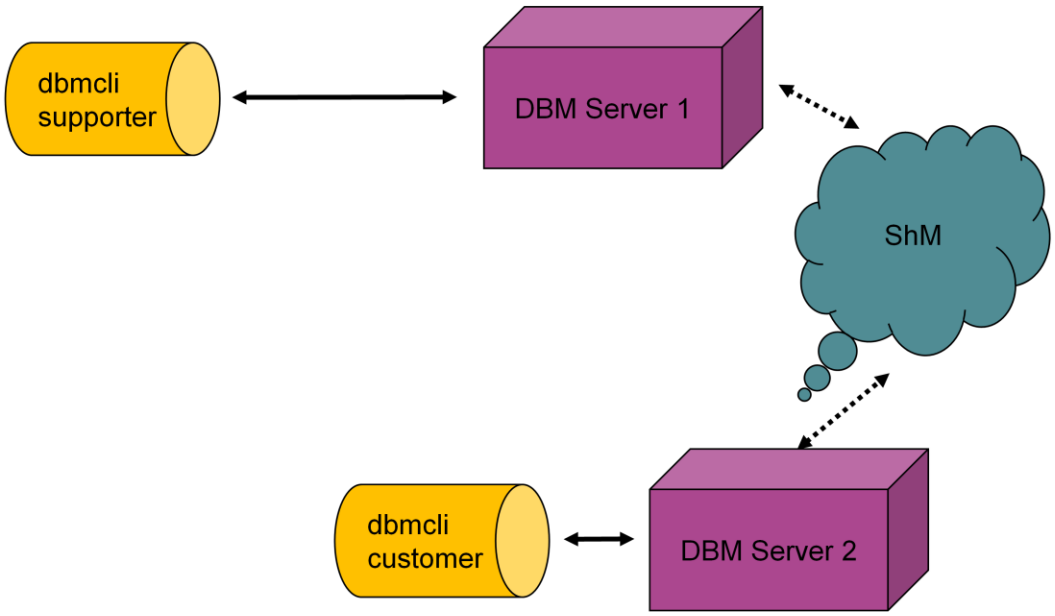
Use cases:

- Supporters can start a long-running command that the customer takes over
- Limited number of connections from SAP to customer is no problem: multiple DBM Server commands can be executed in parallel and be handled with only one real DBM client

This concept is also used internally:
Event Dispatchers and DBM Command Scheduler were ported from separate executables into the DBM Server (hidden commands dbm_schedule and dbm_dispatch_events).

With version 7.7 the concept of background servers was introduced.

Use of Background DBM Server (I)

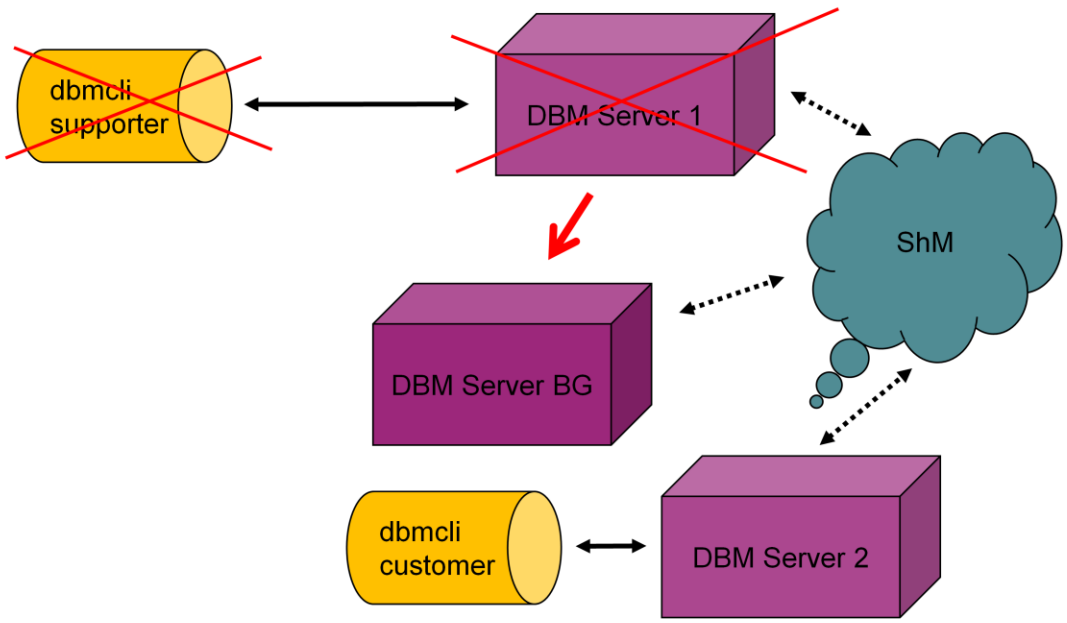


The following two slides desribe a use case for the background server. A supporter has started dbmcli on a remote host and thus also a DBM server on the database server. The DBM server uses the Shared Memory for the database.

If a long running command is issued (like f.e. a backup) the connection may not be interrupted in the meantime. If the connections gets lost also the command stops operation.

The customer could start the command locally but is not familiar with the correct use.

Use of Background DBM Server (II)



The supporter can start a backgound DBM server which executes the backup. The operation can continue even if the supporter disconnects from the database server because the background server is still active. The dbmcli session can be stopped and no other DBM server process is necessary any longer.

Background DBM Server - Commands

- `background_server_execute <bg_server_name> [-no_reply] <command>`
- `background_server_exit <bg_server_name>`
- `background_server_get_reply <bg_server_name> [<skip_bytes>]`
- `background_server_reset <bg_server_name>`
- `background_server_show_status <bg_server_name>`
- `background_server_start <bg_server_name>`
- `background_server_takeover <bg_server_name>`

Firstly the background server has to be started with the **background_server_start** command. **dbm_shm_info** shows the newly generated background server in „idle“ state (waiting for command). With **background_server_execute** a command can be started and the state „ready“ is reached. Before the next command can be started the reply has to be fetched with **background_server_get_reply.**

Now the next command can be issued with **background_server_execute.** Another dbmcli session wants to request the reply. This can be done with the commands **background_server_takeover** (which gives control over the command to the other session) and afterwards **background_server_get_reply .**

background_server_show_status provides information about the status of the background_server. Possible states are:

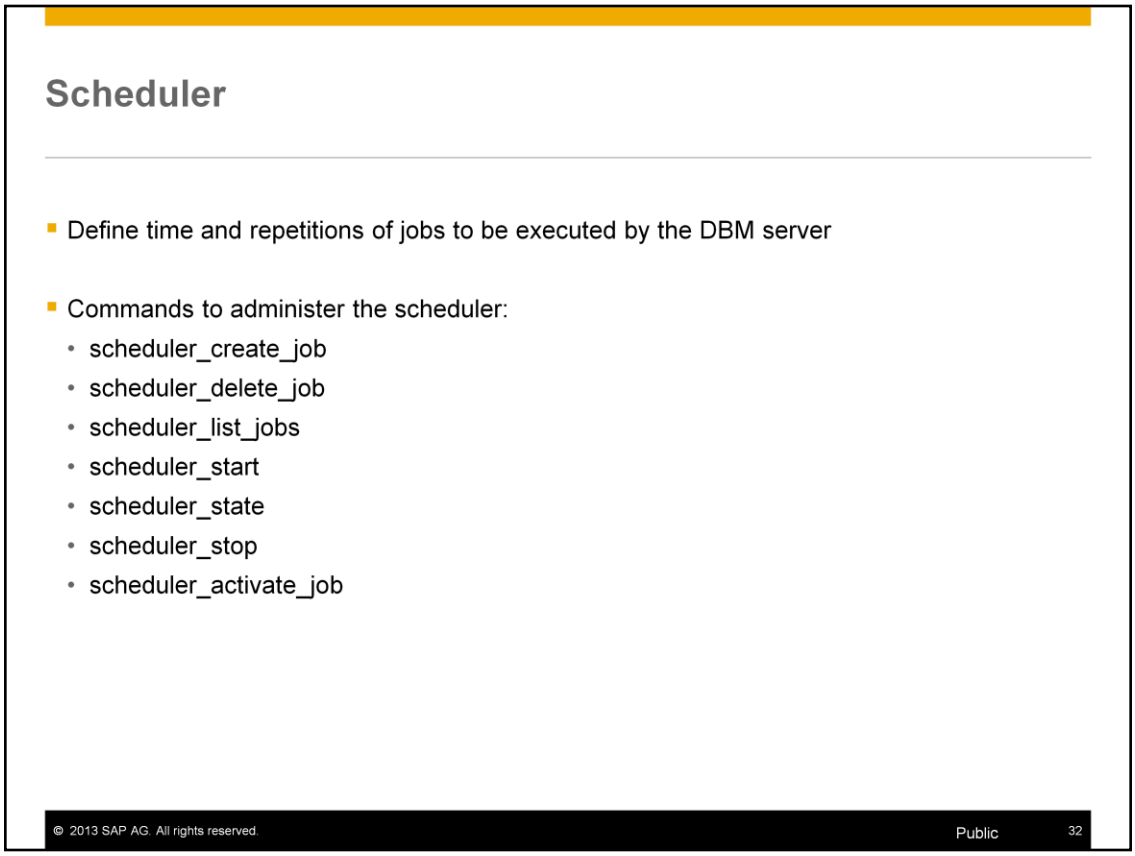
- idle – waiting for command
- running - command is currently active
- ready – command has been executed and result is expected to being fetched
- command available
- starting – the background server is being started; usually short time period
- terminated – the background server has been terminated unexpectedly
- reply read – the reply was being read

Most likely you will see one of the states idle, ready, running.

background_server_exit ends the background DBM server.

Background DBM Server (demo)

- all information to the available background servers, their state and current activity is listed in the output of the dbm_shm_info
- the result of every dbm command started with the background_server_execute command should be picked up with the background_server_get_reply
- for changing the background server state from „ready“ to „idle“ the command background_server_get_reply should be used
- a hanging situation can be resolved with the command background_server_reset



scheduler_create_job

In a scheduler job, you define which DBM command is to be executed and when by Database Manager in the current database.

You can specify when the new job is to be scheduled, and how often or after which job. If you do not specify how often the new job is to be scheduled, the job is scheduled daily at the defined time or every time the job defined as its predecessor is executed.

The DBM operator who created the job is the owner of the job. The system assigns a sequential number for every job created per database.

For creating a job, the operational state of the database is irrelevant. Likewise, it is irrelevant whether the scheduler is on or off.

The new job, however, is only actually executed if the job is active (scheduler_activate_job), if the scheduler is on (scheduler_start), and if you have the server permission to execute the DBM command defined in the job.

scheduler_delete_job

Use this DBM command to delete a scheduler job.

You can delete both active and inactive jobs. Before you do so, however, make sure that the job to be deleted does not have any successors. Otherwise, the DBM command is aborted with an error message. For deleting a job, the operational state of the database is irrelevant. Likewise, it is irrelevant whether the scheduler is on or off.

scheduler_list_jobs

With this command you can display the list of all scheduler jobs.

The job ID, owner, status, time at which the job is to be executed, time at which it was last executed, and the DBM command to be executed are displayed for each job created.

Scheduler - Example

Call Database Manager CLI in session mode, log on as the operator with the password, connect to the database DEMODB.

```
>dbmcli -u operator2,op2 -d DEMODB

dbmcli on DEMODB>
```

Create a scheduler job to stop the database at 6:00 p.m. this evening:

```
dbmcli on DEMODB>scheduler_create_job 18:00:00 db_offline -o

OK 0
```

Create a second scheduler job to start the database at 7:00 p.m. this evening:

```
dbmcli on DEMODB>scheduler_create_job 19:00:00 db_online -o

OK 1
```

Create a third scheduler job to ensure that the operational state of the database is always displayed after the database start, that is, after job 1:

```
dbmcli on DEMODB>scheduler_create_job 1 db_state

OK 2
```

Create a fourth job to create a DBM operator with the user name OPERATOR3, the password op3 and user attributes like the user OPERATOR2. This job is to be executed once, today at 8:00 a.m.

```
dbmcli on DEMODB>scheduler_create_job 08:00:00 "user_create OPERATOR3,op3 OPERATOR2" -once

OK 3
```

Scheduler – Example (II)

```
>dbmcli –u operator2,op2 -d DEMODB
scheduler_list_jobs
OK
OK
ID 0
owner: OPERATOR2
status: active
runs: at: 18:00:00
last run: 2013#05#02 18:00:02
DBM Server PID 2632
return code OK (0)
dbm_command: db_offline
OK
                    . . .
```

```
ID 1
owner: OPERATOR2
status: active
runs: at: 19:00:00
last run: 2013#05#02 19:00:01
DBM Server PID 1652
return code ERR (-24977)
dbm_command: db_online
OK
ID 2
owner: OPERATOR2
status: active
runs: after 1
dbm_command: db_state
```


In this picture you see an example output of command scheduler_list_jobs for the job definitions of the previous slide.

Using Database Events

- An event is a notification from the database system that a certain defined situation has occurred.
- defined by properties such as name and priority
- belongs to an event category
- The values that determine when an event of a certain category is triggered are stored in the system. When this value is reached, the system triggers an event.
- For some categories, there are multiple different property definitions (default values) that trigger an event of that category.
- Using database events, you can automate administration tasks or configure notification options for monitoring your database.

- Prerequisites:
  - server authorization SystemCmd
  - The installation-specific x_server is running on the database computer.
  - A member of the operating system's administrators group has created the file dbm_whitelist.cfg (configuration file in the run directory).
  - The current operating system user can access and execute the program that is to be linked to an event.

Additional info:

Email sending is done via SMTP (RFC 822). The user must provide an SMTP server that „speaks“ this protocol. SAP‘s internal SMTP server is mail.sap.corp (on standard port 25). The DBM Server talks directly to the SMTP server, no JavaMail or other layer is in between, just the network.

User Event Dispatcher

- The user can map an event to the execution of an external program (parameterized with information from the event). Mapped programs are so-called Event Handlers.
- Events can be generated remotely.
- Security consideration:
  - A whitelist (`<rundirectory>/dbm_whitelist.cfg`) must be maintained with software other than MaxDB software. It's a text file that lists all external programs that may be mapped to events.
  - This whitelist must be owned by a member of the local administrator's group (Windows) resp. by root (UNIX).
- Exception:

  The Event Handlers dbmcli and and sdbmail (which is no separate program) may be mapped without being listed in the whitelist.
- The whitelist must exist for the User Event Dispatcher to work.

Event Dispatcher

- DBM Server commands auto_extend and auto_update_statistics use an Event Dispatcher internally, but this is hidden from the user.
- Additionally, the Event Dispatcher is presented as built-in feature of the DBM Server.
  ➔ an Event Dispatcher is a Background DBM Server that executes the command dbm_dispatch events.

- The internal Event Dispatcher and the User Event Dispatcher are two separate DBM Servers with two separate configurations. Hence, the user cannot affect the internal Event Dispatcher.

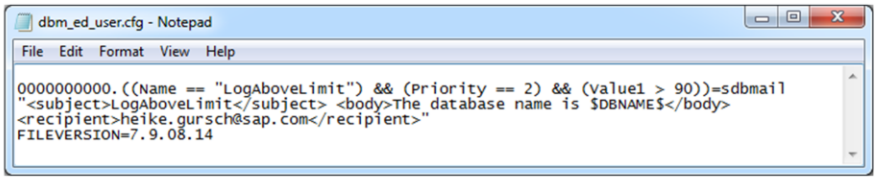
- Managing the User Event Dispatcher:

```
event_dispatcher ADD Name == <value>
                     [Priority == (LOW|MEDIUM|HIGH)]
                     [Value1 (==|>=|<=|>|<) <value>]
                     [Value2 (==|>=|<=|>|<) <value>]
                     Command == <command>       |
                 DELETE <entryID>               |
                 SHOW                           |
                 ON                             |
                 OFF
```

Event Dispatcher Files

configuration of User Event Dispatcher:

<rundirectory>/dbm_ed_user.cfg



configuration of Internal Event Dispatcher:

<rundirectory>/dbm_internal_user.cfg

Whitelist:

<rundirectory>/dbm_whitelist.cfg

In the configuration file directory (directory where configuration file dbm_whitelist.cfg is located) you can check file dbm_ed_user.cfg to get information about your user defined event configuration. The eventing itself is logged in the file dbm.prt.

dbm_internal_user.cfg is the configuration file for the administration of internal events. An internal event dispatcher is always started. Actions like autolog or auto_extend are administered here.

dbm_whitelist.cfg lists all external programs that may be mapped to events. The file must exist but may be empty.

You can test the procedure of defining events using the event *DBM:Test*. An event of category *ThrowDBM:TestEvent* is transmitted, when the DBM command event_create_testevent is executed.

Commands for the Event Dispatcher

event_available	Displaying whether any events were triggered
event_delete	Deactivating an event
event_dispatcher	Managing the event dispatcher function
event_list	Displaying a list of active events
event_list_categories	Displaying event categories and their properties
event_receive	Displaying a triggered event
event_release	Ending an event session
event_set	Defining and activating an event
event_wait	Waiting for an event

List of commands for event dispatcher:

event_available: displays whether events were triggered and can be fetched with event_receive or event_wait.

event_delete: specifies that an event should cease to be active

event_dispatcher: manages the event dispatcher function for a database

event_list: displays the list of all active events

event_list_categories: displays a list of all event categories for which events can be set

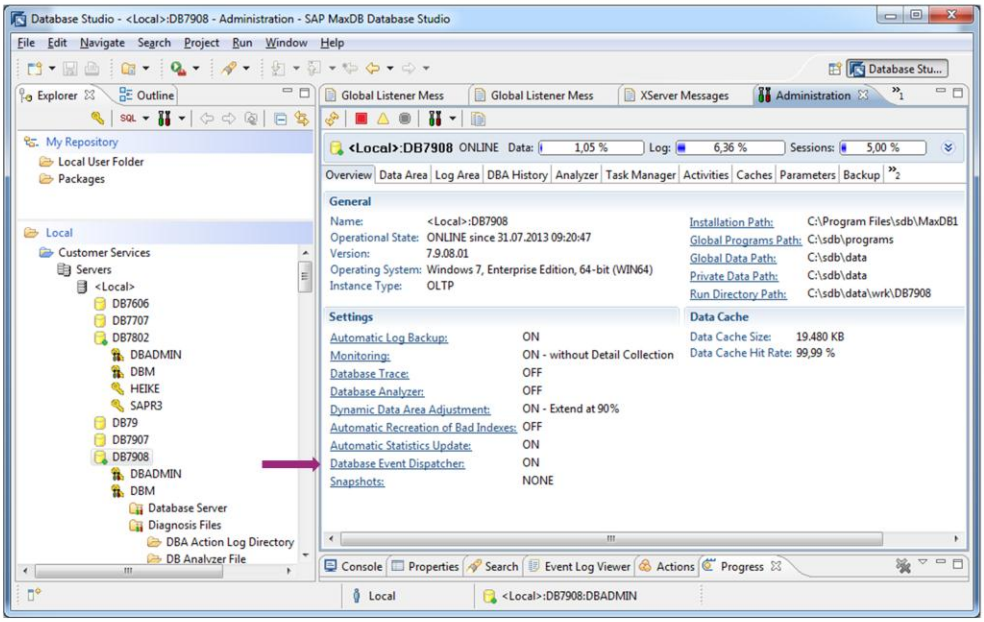
event_receive: fetches an event that has been triggered; repeat this DBM command to fetch additional events.

event_release: ends the event session

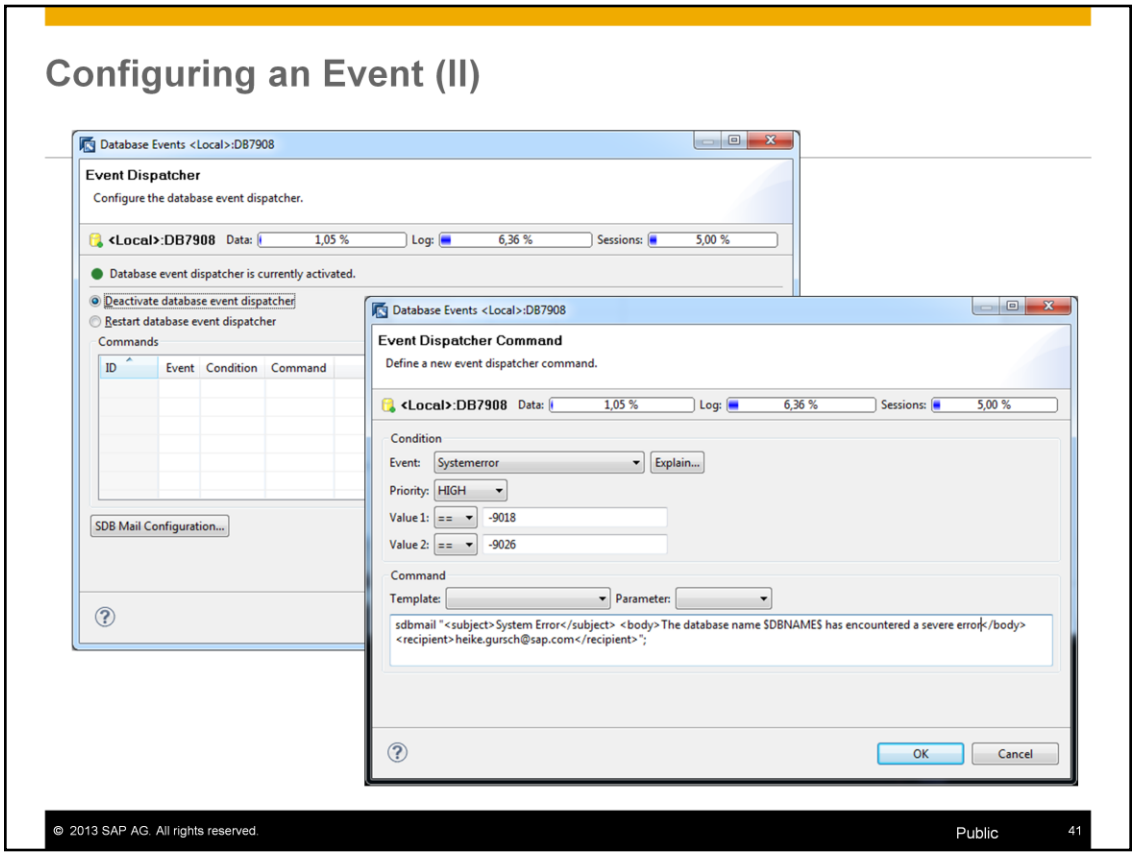
event_set: specifies that the database system should trigger an event as soon as a defined situation occurs

event_wait: fetches a message about the occurrence of an event as well as its data. This is then deleted.

Configuring Events with Database Studio



Firstly you have to start the event dispatcher.



If already events are defined you get a list of it. In the first shown window there is a „New“ Button to define new events.

In the list of possible events you can choose the wished event.

Some examples:

- DBFillingAboveLimit
- DBFillingBelowLimit
- DataCacheAboveLimit
- CheckTableProgress
- UpdateStatWanted
- Systemerror

Afterwards you define the priority and the equal condition.

When you choose „Default Email notification“ as a „Template“ you‘ll get the predefined sdbmail command.

Eventing – Parameters

EventFileName

- Default: knldiag.evt
- name of the event file used for internal diagnostic.
- messages found in this file are database events of any priority
- is only written (in cycles) if EventFileSize is bigger than 0

EventFileSize

- determines the fixed size (in KB) of the event file
- activates/deactivates the writing of knldiag.evt

MaxEvents

- Maximum number of events cached by the kernel for being processed by the Database Manager.

PreservedEventTasks

- A pool of floating service tasks is used to serve event handler and DBAnalyzer-requests
- preserves some of these service tasks for service requests different from event handling

On this slide the kernel parameters are listed that influence the eventing mechanism.

For error diagnosis it is useful to activate the writing of the file knldiag.evt which lists all “fired” events.

The lower limit of 0 disables the caching of any event. It is not recommended to disable the caching of events, if not for minimizing memory consumption since the database manager can only process cached events. If no events are cache the database manager cannot report any events.

A pool of floating service tasks is used to serve event handler- and DBAnalyzer-requests. This parameter preserves some of these service tasks for service requests different from event handling. This way, the available number of service tasks cannot be used up by a single service type.

knldiag.evt - Example

knldiag.evt.txt - Notepad

File Edit Format View Help

```
Date     Time        Index Identifier                  Prio     Value 1    Value 2 Event Text
------------------------------ end of startup part ------------------------------
20130905 123425 0000000001 DatacacheSizeAboveLimit        PrioH 0000001276 0000000020
20130905 123425 0000000002 DatacacheSizeAboveLimit        PrioH 0000001915 0000000030
20130905 123425 0000000003 DatacacheSizeAboveLimit        PrioH 0000002553 0000000040
20130905 123425 0000000004 DatacacheSizeAboveLimit        PrioH 0000003192 0000000050
20130905 123425 0000000005 DatacacheSizeAboveLimit        PrioM 0000003830 0000000060
20130905 123425 0000000006 DatacacheSizeAboveLimit        PrioM 0000004468 0000000070
20130905 123425 0000000007 DatacacheSizeAboveLimit        PrioL 0000005107 0000000080
20130905 123425 0000000008 DatacacheSizeAboveLimit        PrioL 0000005745 0000000090
20130905 123425 0000000009 DatacacheSizeAboveLimit        PrioL 0000006064 0000000095
20130905 123425 0000000010 DatacacheSizeAboveLimit        PrioL 0000006320 0000000099
20130905 123425 0000000011 UpdateStatisticsWanted         PrioL                    UPDATE STATISTICS WANTED

20130905 123425 0000000012 Online                         PrioL 0000000000 0000000000
20130905 123426 0000000013 LogAboveLimit                  PrioL 0000001278 0000000050
20130905 123426 0000000014 LogAboveLimit                  PrioL 0000001917 0000000075
20130905 123606 0000000015 LogAboveLimit                  PrioM 0000002301 0000000090
20130905 123606 0000000016 LogAboveLimit                  PrioM 0000002429 0000000095
20130905 123734 0000000017 BackupResult                   PrioL 0000000000
20130905 123741 0000000018 LogAboveLimit                  PrioL 0000001278 0000000050
20130905 123746 0000000019 LogAboveLimit                  PrioL 0000001917 0000000075
20130905 141238 0000000020 BackupResult                   PrioL 0000000000
20130905 141324 0000000021 LogAboveLimit                  PrioL 0000001278 0000000050
20130905 141329 0000000022 LogAboveLimit                  PrioL 0000001917 0000000075
20130905 141825 0000000023 BackupResult                   PrioL 0000000000
20130905 142010 0000000024 LogAboveLimit                  PrioL 0000001278 0000000050
20130905 142015 0000000025 LogAboveLimit                  PrioL 0000001917 0000000075
20130905 142416 0000000026 BackupResult                   PrioL 0000000000
20130905 142541 0000000027 LogAboveLimit                  PrioL 0000001278 0000000050
20130905 142546 0000000028 LogAboveLimit                  PrioL 0000001917 0000000075
20130905 142646 0000000029 LogAboveLimit                  PrioM 0000002301 0000000090
20130905 142651 0000000030 LogAboveLimit                  PrioM 0000002429 0000000095
20130905 142832 0000000031 BackupResult                   PrioL 0000000000
20130905 142839 0000000032 DBFillingAboveLimit            PrioL 0000004478 0000000070
20130905 142839 0000000033 DBFillingBelowLimit            PrioL 0000004478 0000000070
20130905 142839 0000000034 DBFillingAboveLimit            PrioL 0000004478 0000000070
20130905 142839 0000000035 DBFillingBelowLimit            PrioL 0000004478 0000000070
20130905 142840 0000000036 DBFillingAboveLimit            PrioL 0000004478 0000000070
20130905 142840 0000000037 DBFillingBelowLimit            PrioL 0000004478 0000000070
20130905 142840 0000000038 DBFillingAboveLimit            PrioL 0000004478 0000000070
20130905 142841 0000000039 DBFillingBelowLimit            PrioL 0000004478 0000000070
20130905 142841 0000000040 DBFillingAboveLimit            PrioL 0000004478 0000000070
```

Questions

SAP® MaxDB™ dbmserver

Thank You!
Bye, Bye – And Remember Next Session

Feedback and further information:
http://www.sdn.sap.com/irj/sdn/maxdb

Next Session: 12. November 2013
SAP® MaxDB™ DB Analyzer

SAP® MaxDB™ – Expert Sessions Learning Map (1)

SAP® MaxDB ™ Features	SAP® MaxDB ™ Administration	SAP® MaxDB ™ Problem Analysis
Session 1: Low TCO with the SAP MaxDB Database	Session 2: Basic Administration with Database Studio	Session 5: SAP MaxDB Data Integrity
Session 6: New Features in SAP MaxDB Version 7.7	Session 3: CCMS Integration into the SAP System	Session 14: SAP MaxDB Tracing
Session 8: New Features in SAP MaxDB Version 7.8	Session 11: SAP MaxDB Backup and Recovery	Session 12: Analysis of SQL Locking Situations
	Session 13: Third-Party Backup Tools	
	Session 19: SAP® MaxDB™ Kernel Parameter Handling	

SAP® MaxDB ™ Installation/Upgrade
Session 7: SAP MaxDB Software Update Basics

All Expert Sessions (recording and slides) are available for download
http://maxdb.sap.com/training/

SAP® MaxDB™ – Expert Sessions Learning Map (2)

SAP® MaxDB™ Architecture	SAP® MaxDB ™ Performance
Session 18: Introduction MaxDB Database Architecture	Session 4: Performance Optimization with SAP MaxDB
Session 15: SAP MaxDB No-Reorganization Principle	Session 9: SAP MaxDB Optimized for SAP BW
Session 17: SAP MaxDB Shadow Page Algorithm	Session 16: SAP MaxDB SQL Query Optimization (Part 1)
Session 12: Analysis of SQL Locking Situations	Session 16: SAP MaxDB SQL Query Optimization (Part 2)
Session 10: SAP MaxDB Logging	Session 22: SAP® MaxDB™ Database Analyzer
Session 20: SAP MaxDB Remote SQL Server	
Session 21: SAP MaxDB DBM Server	

All Expert Sessions (recording and slides) are available for download
http://maxdb.sap.com/training/

Thank you

Contact information:

Heike Gursch
IMS MaxDB / liveCache
Heike.Gursch@sap.com

Oksana Alekseious
IMS MaxDB / liveCache
Oksana.Alekseious@sap.com