SAP DB Support Guide

1 Important Tools

1.1 Database Manager (Database Manager GUI, Database Manager CLI and Web DBM, from Version 7.2)

Starting from Version 7.2, Database Manager GUI and Database Manager CLI are the database administration tools.
Database Manager GUI is a graphics variant that can only be installed on Windows hosts, however, it can also administrate UNIX databases.
Database Manager CLI is the command line-oriented variant of the Database Manager.
Web DBM is the web-based variant of the Database Manager.
You can access the Database Manager at www.sapdb.org.

1.1.1 Using Database Manager CLI to Execute SQL Statements

You can use the Database Manager CLI to execute SQL statements. You need Database Manager CLI authorization.

You need to create an SQL session.
```
dbmcli -d <database_name> -u <dbm_user>,<password>
-uSQL <sql_user>,<password>
```

Example:
```
dbmcli -d LCA -u control,control -uSQL sapr3,sap
```

You can then use `sql_execute` to execute an SQL statement:

```
sql_execute select * from messages
```

1.2 DBMGETF

You can use the DBMGETF tool to display specific database files. The following command gives you a list of all files that you can display:
```
dbmgetf -d <database_name> -u <dbm_user>,<password> -l
```

Use the following command to display a specific file:

```
dbmgetf -d <database_name> -u <dbm_user>,<password> -k <file_ID>
```

`<file_ID>` is the identifier shown with option `-l`. If the command is not executed on the database host, then the option `-n` must be used to specify the `<server_name>` of the database host.

1.3 SQL Studio and Web SQL Studio (from Version 7.2)

From Version 7.2, you can use the query tools (SQL Studio and Web SQL Studio) to execute SQL statements for databases.
SQL Studio is a graphical tool that you can install on Windows-based hosts only. However, the tool does allow you to access UNIX databases from a Windows PC.
Web SQL is a web-based tool.
You can access the query tools at www.sapdb.org.

1.4 ireport.py (from Version 7.2)

The tool ireport.py is available from Version 7.2.04. The tool is used to execute SQL statements.

It is located in the directory `<dependent_path>/bin`. The tool ireport.py must be called from this directory, since it also requires other files from this directory.

The tool ireport.py implements the following auxiliary functions:

- **Information on call options**:
  ```
  ireport.py -h
  ```
- **Information on commands**:
  ```
  ireport.py -d <database_name> -u <sql_user>,<password>
  ===>help
  ```

1.5 Check database structure (Verify)

If errors such as BAD PAGES, or other structural errors, occur in a system, and you suspect the cause to be in the hardware, then we recommend that you perform a Check database structure (Verify). These check function checks the structure of the B* trees or the structure of the page chains in liveCache.

You can use the check function in the following ways:

- Database Manager GUI: Use *Check → Database*
- Database Manager CLI:
  `dbmcli -d <database_name> -u <dbm_user>,<password> -uUTL -c util_execute VERIFY`
  `dbmcli -d <database_name> -u <dbm_user>,<password> -uSQL -c sql_execute CHECK TABLE <table_name>`
  As of the Versions 6.2.10 Build 32, 7.2.05 Build 01 and 7.3.00 Build 00 you also have the option WITH SHARE LOCK for CHECK TABLE. If this option is set, a check is made to see whether a LONG actually exists for each LONG surrogate in the table.
  To ensure consistency, a complete SHARE lock must be placed on this table temporarily.
  You cannot use this option with the existing options NO SAVEPOINT or CATALOG.
  The CATALOG option means that only the database catalog is checked, for example, to see whether a changed column actually has a change date record, whether a comment actually exists, and whether any dependent views actually exist. Any errors are corrected implicitly.
  NO SAVEPOINT: No savepoint is triggered after the table analysis.
  You cannot use more than one of these options at any one time.
- XPU
  Check database structure with program XPU

If you performed a Check database structure, then this is recorded in the files `knldiag` or `knldiag.err`; these files also record any errors.

Example:

```
01-06       10:16:25   15               53022 B*TREE    CHECK FILE: 908496
(ROOT)
01-06       10:16:26   15         ERR   54001 I/O       page
00159A47010D0200...000000000020D0200
01-06       10:16:26   15         ERR   54001 I/O       BAD DATA PAGE 1415751
01-06       10:16:26   15         ERR   54001 I/O       on DEVNO 2 DEV_OFFSET
22177
01-06       10:16:26   15         ERR   53021 B*TREE    BAD FILE: 1415751
(ROOT)
```

1.6 Database Console (program X_CONS) (All Versions)

The following documentation on the database console does not claim to be complete. It deals mainly with those commands needed by Development Support to analyze customer problems. To make the documentation clearer, we have not indicated any release dependencies.

The database console (program X_CONS) gives you a quick overview of the operating system resources that the database system is using, the distribution of the database session among the operating system threads, and the status of the active database session. You can also use other functions that are intended mainly for support employees and developers.

Call at Shell Level

```
x_cons <database_name> <command> [<time>
<number_of_repetitions>]
```

In versions lower than 7.2 you can call the console without specifying a path. In Version 7.2 or later, this tool is in the directory `<dependent_path>/bin`, and you must specify this path to call the tool. The `<time>` option specifies the number of seconds after which the command is executed again. You can also specify the `<number_of_repetitions>` of the command in the specified interval.

`x_cons <database_name> help` gives you a complete overview of all available commands.

Call with Database Manager CLI

```
dbmcli -d <database_name> -u <dbm_user>,<password>
db_cons <command> [ -int <time>]
```

As of Version 7.2, you can execute all console commands with Database Manager CLI. The command `db_cons` s used for this.

Available commands

You can use `<command>` to execute the following commands:

Extended Time Measurement Functions

`TIME <ENABLE | DISABLE>:`	Enables or disables special time measurement functions .

Cancel Command or Session

`CANCEL <task_index>:`	Cancels the command that is currently being processed by the task. This means that the cancel flag is set in the database kernel. A prerequisite for canceling the command is that the task is still active in the database kernel, and the cancel flag is queried, or the task is in the state Command WAIT (from 7.x). As of Version 7.x, a set cancel flag is indicated by an exclamation mark after the task activity. The canceled command is then rolled back, which can take some time.
`KILL <task_index>:`	Kills the session of the task. The cancel flag is also set in the database kernel for the kill command. A

prerequisite for killing the session is that the task is still active, and the cancel flag is queried. As of Version 7.x, a set cancel flag is indicated by an exclamation mark after the task activity. The canceled transaction is then rolled back, which can take some time.

`PROCMASK <procmask> [DW|SV|US|GC]`: This function is available on Microsoft Windows only; it must only be used by Development Support in special situations (such as benchmarks). You can use this option to define a fixed assignment between threads and a specific CPU. For example, if there are four processors, then PROCMASK 0001 US assigns the threads that contain user tasks to the fourth CPU. In this way, you can assign threads to specific CPUs in benchmarks. This assignment is deactivated when you restart the database. Do NOT use this option in customer installations.

Statistics and Status Displays:

`SHOW IO`: Displays the I/O accesses to the individual devspaces (volumes). You can use this command to analyze disk activities.

`SHOW AIO (backup only)`: Displays asynchronous I/O accesses (for running backups). You can use this command to analyze backup runtimes.

`SHOW STORAGE`: Displays the memory usage. On Microsoft Windows, the tasks stack is also displayed.

`SHOW TASKS`: Displays all tasks (all types) and their state.

`SHOW ACTIVE [DW|SV|US|GC]`: Displays all active tasks.

`SHOW RUNNABLE [DW|SV|US|GC]`: Displays all tasks that are runnable if they could use the CPU.

`SHOW T_C [DW|SV|US|GC] <Task>`: Displays detailed information on the specified task.

`SHOW VERSIONS`: Displays the version of the database kernel and runtime environment.

`SHOW REGIONS`: Gets information on the different regions.

`SHOW STATE`: Shows the operational state of the database instance, for example, OFFLINE, ADMIN, ONLINE.

`SHOW RTE`: Shows the thread overview, task cluster, and so on.

`SHOW QUEUES`: Displays the queues, for example, tasks that are waiting for cross-thread communication.

`SHOW SUSPENDS`: Shows the cause of any suspended tasks.

`SHOW SLEEP:`	Shows the load on the individual threads. This helps you to find any CPU bottlenecks.
`SHOW THRD_TIMES:`	Shows system and user times for the individual threads. (Microsoft Windows only)
`SHOW ALL:`	Shows all console displays. After a dump, you can find this output in the work directory in the file ***rtedump***.

1.6.1 Process Configuration

Use the database console (program X_CONS).

```
x_cons <database_name> sh rte
```

This shows the distribution of the SAP DB threads among the operating system processes. The special DB threads (coordinator, console, timer, requestor and Dev0) are each realized by a separate operating system thread. The whole database kernel is located in a single Microsoft Windows process.

```
Kernel Threads:
 Thread              Win  State
 Name                Tid

 COORDINATOR       0x748  Sleeping
 TIMER             0x7B4  Sleeping
 CLOCK             0x278  Sleeping
 DEV0              0x7B8  Sleeping
 ASYNC0            0x7C4  Sleeping
 CONSOLE           0x7C8  Running
 REQUESTOR         0x7CC  Sleeping
```

However, multiple database tasks (user task, log writer, utility task, and so on) can be located together in an operating system thread, which is called a UKT (user kernel thread). The SAP DB runtime environment uses internal tasking to administer these database tasks. Internal SAP DB administration takes up less operating system time, and gives you more control over the scheduling and prioritization of individual database sessions.

```
User Kernel Threads:
 Thread          Win  State           Dispatch  Active  Total   Task
 Name            Tid                  Counter   Tasks  Tasks   Cluster
 UKT1          0x7D0  Sleeping              1       1      1   TW
 UKT2          0x7D4  Sleeping              6       1      1   AL
 UKT3          0x7D8  Sleeping             41       0      1   UT
 UKT4          0x7DC  Sleeping             51      12     12   12*SV
 UKT5          0x7E0  Sleeping             10       1      1   1*GC
 UKT6          0x7E8  Sleeping             57       9      9   TI,8*DW
 UKT7          0x7E4  Sleeping             68       0     50   50*US
```

The database parameter MAXCPU is normally used to distribute the tasks automatically to the UKTs; however, the (support) database parameters _TASKCLUSTER_01, _TASKCLUSTER_02 and _TASKCLUSTER_03 can also be used to modify this distribution.
The dispatch counter tells you how often the database tasks have had control of the UKT.

Abbreviations of the Database Tasks in TASKCLUSTER

Abbreviation	
tw	Trace writer; writes the database trace (kernel trace) and dump, if necessary.
al	Log writer for logging
ut	Utility task for administration tasks (start backup, recovery, and so on).
sv	Server tasks for backup I/O, as well as special tasks such as creating parallel indexes, prefetching (liveCache), Check database structure (verify)
ti	Timeout task for timeout monitoring
dw	Pager (Data writer) for cache monitoring and asynchronous swapping, as well as savepoint I/O, devspace (volume) balancing
us	User tasks for executing SQL statements
cs	Converter scanner for filling the PNO pool (if necessary) – only in releases before 7.4.

1.6.2 Idle UKT Times (UKT Sleep Statistics)

Use the database console (program X_CONS).

```
x_cons <database_name> show sleep[time]
```

Displays the load on the UKTs (user kernel threads).

	Idle	dd:hh:mm:ss		I/O Wait	dd:hh:mm:ss		RQlen
UKT2	255673	09:58:46	60.46%	0		0.00%	0
UKT3	6627	16:29:44	99.94%	0		0.00%	0
UKT4	1007977	16:15:43	98.52%	33033	13:08	1.33%	0
UKT5	1464884	10:06:48	61.27%	1449353	06:19:00	38.27%	0
UKT6	5935928	01:27:15	8.81%	5346924	11:00:13	66.66%	0
UKT7	4821734	03:39:07	22.13%	3756545	09:58:03	60.39%	0

Pay special attention to the columns *Idle* and *Rqlen* (run queue length). Any growth in the numbers in the *Rqlen* column indicates that database tasks within a UKT are waiting for the CPU which is probably used by another task within the same UKT. You can use the command `x_cons show runnable` to display these waiting tasks. Find out whether internal SAP DB dispatching can solve this situation. If the idle counter also does not change (the UKT is always working), then this indicates a long running DB procedure, that does not give control back to the database kernel, and cannot be dispatched. This can cause performance problems, since all other tasks that run in the same UKT cannot work.

1.6.3 Task Activity and Task State

Use the database console (program X_CONS).
`x_cons <database_name> show active [DW|SV|US|GC] [<time>]`
Gives you an overview of the current activity of the active database sessions and special tasks. Only those tasks are displayed that are currently be processed by the database kernel (no display if a user task is waiting for a new command, status: Command Wait). Special tasks are displayed only when they are not in a sleep or suspend state (vsleep, vsuspend).

```
ID  UKT UNIX  TYPE    APPL  State          timeout Region  Wait UKTsleep
        tid           pid                  sec     cnt/try
T2  14    12 ALogWr.    -1 IO Wait (W)     0       0   0    0   12765
(r)
T12 16    30 DataWr.    -1 IO Wait (W)     0       0   0    0   12765
(r)
T23 18    31 User    11265 Running         0       32  9    0   12765
(r)
T24 18    31 User    11258 IO Wait (R)     0       0   0    0   12765
(r)
T25 18    31 User    11258 Vsuspend        0       0   0    0   12765
(r)
T26 18    31 User    11268 Runnable        0       0   0    0   12765
(r)
T27 19   116 User    15218 DcomObjCalled   0       102 0    0
225918(r)
T28 19   116 User  * 49544 DcomObjCalled   0       102 0    0
225918(r)
T29 19   116 User  * 43904 Command wait    0       0   0    0
225918(r)
T30 19   116 User  * 43352 Vsuspend (232)  0       102 0    0
225918(r)
T31 19   116 User  * 35700 Vsuspend (227)& 0       102 0    0
225918(r)
```

ID: Specifies the task ID

UKT: Specifies the number of the user kernel thread

UNIX tid : Specifies the Unix process ID of the user kernel thread

TYPE: Indicates the type of the task

Task Type	**Meaning**
Timer task	The **timer task** is used for handling all types of timeout situations (such as session timeouts and lock request timeouts).
Log writer	The log writer is responsible for writing the before and after images to the log devspaces (volumes) (before images in SAP DB versions older than 7.4 only).
Trace writer	SAP DB gives you the option of writing a special log, the database trace, for analysis purposes. The trace writer becomes active when the trace is flushed. The active tasks write the trace data to a buffer. The trace writer writes the data from the buffer to the file ***knltrace***.

Task Type	Meaning
Converter scanner	The **converter scanner** (versions lower than 7.4 only) quickly supplies the addresses of empty data pages. It reads the converter and enters the numbers of the empty data pages in the Pnopool.
Utility task	The **utility task** is reserved exclusively for the administration of the database instance. Since only one utility task exists for each database instance, administration tasks cannot be performed in parallel. This removes the possibility of any conflicts. One exception to this are the automatic log backups. These can be performed in parallel with other administration tasks.
Pager (Data writer)	**Pagers (Data writers)** are responsible for writing data from the data cache to the data devspace (volume). They become active when a savepoint or checkpoint (in releases older than 7.4) is performed, during devspace (volume) balancing, and during swapping between savepoints. The system calculates the number of data writers. This number primarily depends on the size of the data cache and the number of data devspaces (volumes).
Server task	Primarily, **server tasks** are used to back up data. Some server tasks read from the data devspaces (volumes); others write to the backup medium. The CREATE INDEX statement instructs the server tasks to read the table data in parallel from the data devspaces (volumes). The system calculates the number of server tasks needed in the configuration of the database instance automatically from the number of data devspaces (volumes) and the number of backup devices (see above).
User task	Precisely one **user task** is assigned to each user or application process/thread at logon/startup. The maximum number of available user tasks is determined by the database parameter MAXUSERTASKS. This parameter also restricts the number of user sessions that can be logged on to the database system simultaneously. The database parameter _MAXTASK_STACK determines the memory usage of the user tasks.

APPL pid:
Shows the process ID of the application program linked to the task. An asterisk (*) before the PID indicates that the process ID is on a separate host (application server), and it is being accessed remotely.

State:
Indicates the state of a task.

Status	Meaning
Running	The task is active and is using the CPU. (Systems with one CPU can only have one 'running' task at a time. However, if x_cons shows two 'running' tasks, then this is caused by unprotected access, which has no time restrictions.)
Runnable	The task could run now, but another task is using the CPU. Internal SAP DB dispatching suspends the task due to long runtime, or the prioritization of another task.
IO Wait (W)	The task waits for I/O, which writes a page to disk. If a user task is displayed in this state, then swapping might exist. The user task must displace data from the cache before it can read new data. Try to avoid this situation, since it requires two I/Os for one read, which is bad for performance. If this situation occurs repeatedly in a customer system, you must perform a detailed performance analysis to find out the cause of the swapping.
IO Wait (R)	The task waits for I/O, which reads a page from the devspace (volume) and writes it to the

Status	Meaning
	cache. Each page requested by the application must first be loaded into the cache. IO WAIT (R) is not unusual after starting the system, however, you should be able to retain as much data as possible in the cache in a running system, and this is what you should aim for in the liveCache environment.
Vwait	The task has requested an SQL lock that it cannot have. It is waiting for the lock.
Vbegexcl	The task is waiting for a region (synchronized memory access), or enters a region.
Vendexcl	The task exits the critical region. Comment: Conflicts can also occur when the critical region is released, since the release must also be synchronized. If, at the same time, a task wants this critical region in another UKT, then conflicts might occur (see section 5.8.6).
Vsuspend	The task is waiting for a B* tree lock (for a very short period of time) or for LOG I/O (user tasks). Usually, a number appears after the Vsuspend status that gives the reason for the suspend. You can use the console command *show susp* and the number to find out the reason for the wait situation from the list. The system explicitly wakes any tasks in this state.
Vsleep	The task is in a short wait situation. It starts running again after a predefined period.
Vopmsg	A message is written to the file ***knldiag***, ***knldiag.err*** and/or ***opmsg(n)***.

Timeout sec:
If the task is in a state in which a timeout is active, such as vwait, then this column records the seconds until the timeout ends.

Region cnt/try:
Caches can consist of one or more critical regions. For example, the data cache has between eight and 64 critical regions, depending on its size. Semaphores control the access to these different critical regions.
If a task accesses a critical region, then the semaphore locks this region for all other tasks.

cnt:
Displays the number of times a critical region has been accessed since the task has been running.

try:
Number of the queried or held critical region.

UKTsleep:
Number of semaphore waits; dependent on UKT (IDLE counter).

1.6.4 Debug Option

Use the database console (program X_CONS).
`x_cons <database_name> deb <debug_level>`

This command activates enhanced console output for the detailed analysis of particular problem situations.
In lock situations, such as when `x_cons <database_name> show act` shows a large number of tasks in the Vwait or Vsuspend status, it might be a good idea to make a

more precise analysis of the conflicts. In this case, debug level 3 displays extra information.
When you activate debug level 3 (originally implemented for internal SAP DB developers only), additional information is displayed for database tasks in wait and lock situations. The information is displayed in columns whose headers do not indicate the new meanings. Also, the layout and meaning of the information can be changed during an SAP DB version.
Tasks have a higher priority in the SAP DB kernel when they lock a resource that is requested by another task. This means that the task is processed with priority to release the resources needed by the other task. This prevents queues forming before locked objects (quickest possible removal of the lock). You can use the xparam parameters to change the priorities. Prioritization is deactivated in minimum configurations, such as laptops.

ID	UNIX tid	TYPE	APPL pid	State	timeout	Region cnt	Region try	Wait sec	UKTsleep
T29 (r)	31	User	9909	sus-RLT (060)	0	0	30	5054	167039
T30 (r)	31	User	9897	sus-WLT (**058**)	**2** 0	0	**34**	**5054**	167039
T33 (r)	31	User	9921	Runnable	0	0	4	0	167039
T34 (r)	31	User	9994	Running	**2** 0	2614	41	0	167039
T35 (r)	31	User	9904	sus-RLT (060)	0	0	30	5054	167039
T41 (r)	31	User	9916	Running	0	64	41	0	167047
T42 (r)	31	User	9900	Vbegexcl	0	0	27	0	167047
T43 (r)	31	User	9921	Vwait	0	0	**44**	34256	167047
T44 (r)	31	User	9894	IO Wait (R)	**1** 0	0	5	34256	167047

As well as the task status (running, I/O wait, and so on.), a figure in the column *State* shows you the type of conflict for the prioritized (locking) tasks (1: task has SQL lock; 2: task has B* tree lock; or 3: task has both SQL and B* tree lock).

Task T30 is suspended and is waiting for the tree lock of root 5054, which is being held by task 34.
The action that task T30 wants to perform is shown behind the *State* column in parentheses, for example, (058). You can determine this action with the x_cons command `show susp`.
Task T44 is prioritized, since it has an SQL lock. Task T43 has the state vwait and is waiting for an SQL lock that is held by task T44 (region try 44).

1.6.5 Reasons for Task Suspends

Use the database console (program X_CONS).

```
x_cons <database_name> show susp
```

Displays the list of possible causes for a suspend. The number from the list is used to make an assignment to the program function. You can use the program source code of the appropriate release to analyze the reason for the suspend.

```
List of suspend reasons:
========================

total suspends:  119657

No-Work  (049) :       1 (   0.00% ) Trace_Manager::WaitForWork
InitState(052) :       8 (   0.01% ) b12data_writer
No-Work  (053) :      32 (   0.03% ) b12data_writer
Vsuspend (054) :       2 (   0.00% ) bd12wait_for_dw_reply
Vsuspend (204) :       2 (   0.00% ) kb39write_wait
Vsuspend (211) :      24 (   0.02% ) k90sreceiver_server
Vsuspend (212) :       2 (   0.00% ) kb90parent_wait
No-Work  (228) :  119572 (  99.93% ) bd91ObjGarbageCollector
NoRedoJob(231) :       7 (   0.01% ) Log_RedoTrafficControl::ExecuteJobs()
LogIOwait(234) :       2 (   0.00% ) Log_Queue::UserTaskEOTReady
No-Work  (235) :       3 (   0.00% ) Log_Writer
InitState(242) :       1 (   0.00% ) Log_Writer
SVP-wait (243) :       1 (   0.00% )
```

1.6.6 Task Detail

Use the database console (program X_CONS).

`x_cons <database_name> show t_c t<task_index>`

Shows detailed measurements for individual database tasks. In this way, you can, for example, monitor the DB activity of an application while it remains connected to a database task (no permanent release/connect).

You can also use

`x_cons <database_name> time enable`

to activate a precise time measurement of different database states. Depending on the operating system, this time measurement costs between 1% and 5% performance. Much of the output of the `show t_c` function was developed exclusively for developers, however, some of the values are of more general interest in special situations.

```
-------------------  T25   User      ( pid = 23163  ) ----------------------
 dispatcher-cnt: 127292                               command-cnt   : 30477
 total_excl-cnt: 9110558                              self_susp-cnt : 433
 dev_write_io  : 19         dev_write_pg  : 19        avg_dev_wr_tm : 0.0895
 state_vwait   : 11                                   avg_vwait_time: 4.1446
 state_vsusp   : 682                                  avg_vsusp_time: 0.0684
 rcv_rpl_count : 2296       rcv_rpl_long  : 46        avg_rcv_rpl_t : 0.1677
 rpl_rcv_count : 2296                                 avg_rpl_rcv_t : 0.0222
 prio_total    : 60                                   prio_from_oth : 0
 prio_com_cnt  : 733        mod_own_com   : 8         mod_other_com : 0
 prio_rav_cnt  : 21         mod_own_rav   : 0         mod_other_rav : 0
-------------------------------------------------------------------------
```

Meaning of the Columns

dispatcher-cnt	Count of how often the task passed control to the UKT dispatcher, because it could not run, its time had run out, or another task was prioritized.
total_excl-cnt	Number of critical region accesses
command-cnt	Communication count between application and kernel
self_suspend-cnt	Number of task suspends where the task could run, but still passed control to another task in the same UKT.
<dev/self>_<read/write>_io	Number of I/Os via UKT (self) or DEV thread (dev)
<dev/self>_<read/write>_tm	Duration of an I/O via UKT (self) or DEV thread (dev)
state_vwait	Number of waits on SQL locks.
avg_wait_time	Average wait time for an SQL lock
avg_rcv_rpl_t	Average processing time for an SQL statement in the database kernel
rcv_rpl_long	Number of SQL statements with a processing time of more than one second

1.6.7 Critical Regions

Use the database console (program X_CONS).

```
x_cons <database_name> show region
```

Displays critical region access statistics.

```
Index  Region    Owner    Get-Cnt    Tas-Cnt      Coll.    Waits Excl   Coll %
    1  BACKUP                  0          0          0        0    0   0.00 %
    2  BUFWRTR                 0          0          0        0    0   0.00 %
    3  DATAWRTR              102          0          0        0    0   0.00 %
    4  CONFIG                  5          0          0        0    0   0.00 %
    5  CONV1                  36          0          0        0    0   0.00 %
    :
    :
   13  DATACACH                0          0          0        0    0   0.00 %

   :
   55  TREE1                 392          0          0        0    0   0.00 %
   :
   63  SPLIT1                  4          0          0        0    0   0.00 %
   :
   71  DATA1                 503          0          0        0    0   0.00 %
   :
   79  TRANS1               2173          0          0        0    0   0.00 %
   :
   87  TAB9                 2174          0          0        0    0   0.00 %
   :
   95  ROW17                2154          0          0        0    0   0.00 %
   :
```

Meaning of the Columns

Owner:	The TASK ID appears here if the critical region is being held by a task. If tasks are waiting for this critical region in the region queue, then a list of these tasks appears in the following lines.

In the following example, all named tasks are waiting for the garbage collector region, which is being held by the task T70.

```
Index  Region    Owner    Get-Cnt    Tas-Cnt      Coll.    Waits Excl   Coll %

13     GARBAGE   T70      2072493         47        473      602    0   0.0
                                      waiting: T119
                                      waiting: T208
                                      waiting: T120
                                      waiting: T156
                                      waiting: T118
                                      waiting: T136
                                      waiting: T114
                                      waiting: T1
                                      waiting: T185
                                      waiting: T186
```

Get_cnt: Specifies the number of times this critical region was accessed.

Tas_cnt: Specifies the number of conflicts that occurred when the region administration function was accessed (simultaneous entry/exit of tasks running in different UKTs).

Coll.: Specifies the absolute number of conflicts in a critical region. A conflict occurs when a task requests a critical region that is already being held by another task.
Comment: If the conflicting task is not entered in the region queue immediately, and instead tries to access this critical region up to MP_RGN_LOOP (parameter) times, then each of these failed attempts counts as a conflict. This is how conflict rates of well over 100% can occur. (1 get_cnt can result in MP_RGN_LOOP conflicts).

Waits: Specifies how often tasks have registered as waiting in the region queue. For MAXCPU= 1 (so that MP_RGN_LOOP=1), this value is the number of conflicts.

Excl.: Flag that indicates whether the lock is set for region administration.

Coll %: Specifies the conflict rate in this critical region.

Version 7.4.03 implements new x_cons output for AWE and task hopping. However, we did not have details about this output at the time of writing.

1.7 X_DIAG(NOSE) (All Versions)

Version 6.2 includes the X_DIAGNOSE tool.
Windows: `x_diag` in the directory `%DBROOT%\bin`
UNIX: `x_diagnose` in the directory `$DBROOT/bin`
You can call this tool without specifying a path.

As of Version 7.2.04, the tool is located in the directory `<dependent_path>/bin`. Since this directory is not in the path, you must call the tool by specifying the path.

If you call X_DIAGNOSE with the options
`-d <database_name> -u <dbm_user>,<password>`
then additional functions become available. The guide specifies whether you need to activate these options.

For more detailed information on using X_DIAGNOSE, see chapters 7.8 and 7.9.

1.8 XINSTINFO (from Version 7.2.05)

The tool XINSTINFO is available as of Version 7.2.05 Build 04 or Version 7.3.00 Build 06; this tool displays all important information on the installation directories of a system. You can call the tool without specifying a path.

- **Call without parameters**:
  `xinstinfo`

  Example Output:
  ```
  IndepData         : /sapdb/IndepData
  IndepPrograms     : /sapdb/IndepPrograms
  ```

- **Call with Database Name**:
  `xinstinfo <database_name>`

  Example Output:
  ```
  IndepData         : /sapdb/Data
  IndepPrograms     : /sapdb/Programs
  InstallationPath  : /sapdb/MUT/db
  Kernelversion        : KERNEL    7.2.5    BUILD 004-000-
240-291
  Rundirectory         : /sapdb/Data/wrk/MUT
  ```

2 Important Analysis Files

Newer database versions back up the following diagnosis files automatically. You do not need to back up these files explicitly.

from 72.05.00 (Windows), 72.05.01 (UNIX), 73.00.05, 74.00.00:
`knldiag, knltrace, knldump, rtedump, *.dmp, *.buf, *.stm`

from 74.00.00 (UNIX) also:
`core`

This function makes sure that important diagnosis files are not overwritten when the database is restarted after a crash.
If the database recognizes that it is being restarted after a crash, then the necessary files are backed up to a directory with the following naming convention:

`<database_name>_<date>_<time>`
for example: `DB72_20001114_12-09-45`

The backed up diagnosis files are deleted from the original directory.
The backup directory is under the directory DIAG_HISTORY_PATH (which must be configured) and is known as **history** from now on.

You can also configures the number of histories (DIAG_HISTORY_NUM). If you exceed this number of histories, then the oldest history is deleted when a new backup is made.
The database can still be restarted if a backup cannot be made correctly.
Deleting a database instance also deletes all histories for the database. If this empties the directory DIAG_HISTORY_PATH (because it does not contain any histories of other databases), then it is also removed.
When you back up the `core` file to another files system, remember that the file is expanded to its full size, which can be very large. As of Version 7.4.02 Build 04 the default parameter SUPRESS_CORE=YES specifies that the core is not written.

Parameter:

- DIAG_HISTORY_NUM:
  Value 1 or greater; No space check is made on the disk for this value
  Default: 2
- DIAG_HISTORY_PATH:
  The path name can have a maximum of 64 characters.
  Default: <RUNDIRECTORY>\DIAGHISTORY

2.1 knldiag, knldiag.old and knldiag.err

These files contain status and error messages from the database kernel.

knldiag:

`knldiag` is the current file. This file is recreated with the configured size (database parameter KERNELDIAGSIZE) when the database is restarted. The file is written to the run directory of the database (database parameter RUNDIRECTORY), unless the path of the file has been changed (database parameter KERNELDIAGFILE).

The start of the file lists messages about the start of the database in the state COLD/ADMIN – this part of the file is separated from the rest of the file by a broken line and is not overwritten in cycles. It then lists messages from the running database; these messages are overwritten in cycles. For more detailed information about the messages in the start phase, see section 13.

As of Version 7.2.05 Build 15 the stack back trace is also logged in this file after a crash.

Example of knldiag Version 6.2.10 on Windows NT:

```
------------------------------------------------------------------------------
Date  Time       TID(hex) Typ MsgID Label    Message-Text
------------------------------------------------------------------------------
10-19 10:01:43       0xFC     50000 TCLUSTER
tw;al;ut;2000*sv;10*ev;ti,100*dw,sn,rc,cs;30000*us;compress
10-19 10:01:43       0xFC     50001 TCLUSTER number of ' DW':   8
10-19 10:01:43       0xFC     50001 TCLUSTER number of ' EV':   0
10-19 10:01:43       0xFC     50001 TCLUSTER number of ' US':   5
10-19 10:01:43       0xFC     50001 TCLUSTER number of ' SV':  13
10-19 10:01:43       0xFC     18313 INFO     Starting SERVERDB:   'DEM'
10-19 10:01:43       0xFC     18314 INFO              SERVERNODE: 'P33439'
10-19 10:01:43       0xFC     18315 INFO              Process ID: 0xe4
10-19 10:01:43       0xFC     18317 INFO              Date:       00-10-19
10-19 10:01:43       0xFC     18316 INFO              Owner:      'SYSTEM'
10-19 10:01:43       0xFC     18319 INFO              Number of Processors:         1
10-19 10:01:43       0xFC     18320 INFO              Max virtual memory:        2047 MB
10-19 10:01:43       0xFC     18321 INFO              Total physical memory:      159 MB
10-19 10:01:43       0xFC     18322 INFO              Available physical memory:   72 MB
```

```
10-19 10:01:43       0xFC     18323 INFO              Kernel shared memory size:    7 MB
10-19 10:01:43      0x112     18257 TASKING  UKT started, TID:0x112
10-19 10:01:43      0x1F1     18257 TASKING  UKT started, TID:0x1F1
10-19 10:01:43      0x106     18257 TASKING  UKT started, TID:0x106
10-19 10:01:43      0x1FB     18257 TASKING  UKT started, TID:0x1FB
10-19 10:01:43      0x130     18257 TASKING  UKT started, TID:0x130
10-19 10:01:43      0x144     18257 TASKING  UKT started, TID:0x144
10-19 10:01:43      0x144     54003 dynpool  NUM DATAWRITER         :      8
...
10-19 10:01:43      0x144     54003 DYNPOOL  DYNP_B11_FBM_STRUC     :   3212
10-19 10:01:43      0x144     18230 VERSION  'KERNEL    6.2.10   Build 025-000-044-836'
10-19 10:01:43      0x144     18230 VERSION  'NT/INTEL  6.2.10   Build 025-000-044-836'
10-19 10:01:43      0x144     54003 dynDATA  DYND_K57_KB_PAGES      :     11
...
10-19 10:01:44      0x144     53040 I/O      DATAWRIT first datacache: 7
10-19 10:01:44      0x112     18245 DEVIO    Attaching devspace 'knltrace'
10-19 10:01:46      0x16F     18243 DBSTATE  I/O thread for 'knltrace' started
10-19 10:01:46      0x112     54003 dynDATA  DYND_B12_VTRACE        :      2
10-19 10:01:47      0x1C3     18231 DBSTATE  SERVERDB is ready
============================ begin of write cycle ============================
10-19 10:02:26      0x106     18263 CONNECT  Connect req. (T7, Node:'', PID:0x1BD)
10-19 10:02:26      0x106     18245 DEVIO    Attaching devspace 'SYS_001'
10-19 10:02:26      0x110     18243 DBSTATE  I/O thread for 'SYS_001' started
10-19 10:02:26      0x106     18247 DEVIO    Single I/O attach, 'SYS_001', UKT:3
10-19 10:02:26      0x106     18246 DEVIO    Detaching devspace 'SYS_001'
10-19 10:02:26      0x110     18244 DBSTATE  I/O thread for 'SYS_001' stopped
10-19 10:02:26      0x20E     18248 DEVIO    Single I/O detach, 'SYS_001', UKT:3
10-19 10:02:26      0x106     18245 DEVIO    Attaching devspace 'diskl01'
10-19 10:02:26      0x113     18243 DBSTATE  I/O thread for 'diskl01' started
10-19 10:02:26      0x106     18247 DEVIO    Single I/O attach, 'diskl01', UKT:3
10-19 10:02:26      0x106     18246 DEVIO    Detaching devspace 'diskl01'
10-19 10:02:26      0x113     18244 DBSTATE  I/O thread for 'diskl01' stopped
10-19 10:02:26      0x20E     18248 DEVIO    Single I/O detach, 'diskl01', UKT:3
10-19 10:02:26      0x106     18282 CONNECT  Connection released, T7
10-19 11:28:25       0xFC     18241 DBSTATE  SERVERDB is being stopped
10-19 11:28:25      0x112     18247 DEVIO    Single I/O attach, 'knltrace', UKT:1
10-19 11:28:26      0x112     18249 TASKING  Releasing tracewriter
10-19 11:28:26       0xFC     18320 TASKING  Tracewriter termination timeout: 1200 sec
10-19 11:28:26       0xFC     18242 DBSTATE  SERVERDB 'DEM' has stopped,
   Version:    'KERNEL    6.2.10   Build 025-000-044-836'
--------------------------- current write position ---------------------------
```

knldiag.old:

When the database is restarted, the file `knldiag` is renamed as `knldiag.old`. This means that an older version of this diagnosis file always exists. If you restart the database repeatedly, it may be the case that you cannot access important error messages, since the latest restart has already overwritten the file `knldiag.old` with the current `knldiag`. **IMPORTANT**: You must back up these files before restarting the database after a crash. (See section 2 about version dependencies.)

knldiag.err:

The file `knldiag.err` is not overwritten. This file logs all error messages . If the files `knldiag` and `knldiag.old` no longer contain an error, then you can find it in `knldiag.err`. However, you can only see the error message itself; the file does not log the surrounding messages.
New messages are added to the end of the file, which means that it can become very big. On Windows, `knldiag.err` logs the message '`Starting`' for each restart in the state COLD/ADMIN. On UNIX, this happens only from Version 7.2.05.

Example of knldiag.err Version 7.3.00 on Windows NT:

```
07-01 19:44:27       0x40 ERR 53187 B*TREE   BD53: predsep.k > sep.k: 0
07-01 19:44:27       0x40 ERR 52050 SHUTDOWN column/invalid_leaves_st
07-01 19:44:27      0x13A ERR 52050 SHUTDOWN partial_rollback/shutdown
07-01 19:44:27       0x40 ERR 52050 SHUTDOWN *** EMERGENCY ***: 9140
07-01 19:44:27      0x13A ERR 52050 SHUTDOWN *** EMERGENCY ***: 1900
07-01 19:46:07                              --- Starting ---
07-01 19:46:56       0xDE ERR 55012 FBM     invalid devno(26) or offset(1015382)
07-01 19:46:56       0xDE ERR 51080 SYSERROR -9050 Message not available
07-01 22:37:31                              --- Starting ---
07-01 22:37:51     0x1DD ERR 55012 FBM     invalid devno(26) or offset(1015382)
07-01 22:37:51      0x1DD ERR 51080 SYSERROR -9050 Message not available
07-02 09:50:14                              --- Starting ---
```

2.2 dbm.* or control.*

The tools Database Manager GUI and Database Manager CLI (Version 7.x), as well as CONTROL (Version 6.x), also write log files. You can find them in the run directory of the database. If you encounter any problems, search the files `dbm.prt` and `*.utl` for error messages.

File	Content
*.knl	Backup history
*.utl	Administration commands sent to the database kernel, such as backup and restart commands
*.ins	Log for system table load
dbm.prt	Commands sent by Database Manager (CLI or GUI) to the DBM server
control.log	Log of the actions executed by CONTROL
dbm.mmm or control.med	Definition of backup media
dbm.ebp	Log of actions with external backup media
dbm.ebf	Command ID and backup label, as well as external backup ID (EBID)
dbm.mdf	Command ID and backup label, as well as the defined media at the time of the backup

2.3 knltrace

This file can contain important information about the causes of a crash. It is written in binary format, and is reinitialized when the database is restarted. This means that it is very important that you back up this file after a crash (and BEFORE the next restart) (see section 2 on version dependencies).
Data is also written to this file when writing database trace is activated.
The file is evaluated with the tool X_DIAG(NOSE) or XKERNPROT (see section 3.1)
The size of the file `knltrace` depends on the parameter KERNELTRACESIZE (Versions older than 7.2.05) or on various other parameters (Version 7.2.05 and later):

KERNELTRACESIZE is a read-only parameter that results from the following calculation:

```
CALC 1  \
     TRACE_PAGES_TI + \
     TRACE_PAGES_GC _MAXGARBAGE_COLL * + \
     TRACE_PAGES_AL + \
     TRACE_PAGES_DW _MAXDATAWRITER * + \
     TRACE_PAGES_US MAXUSERTASKS * +  \
     TRACE_PAGES_UT + \
     TRACE_PAGES_SV MAXSERVERTASKS * + \
     TRACE_PAGES_EV _MAXEVENTTASKS * + \
     TRACE_PAGES_CS + \
     TRACE_PAGES_BUP _MAXBACKUPTASKS * +
```

The TRACE_PAGES_.. information documents (in pages) how much trace space this task type (timeout TI, user US, ...) assigns to each task. The defaults are:

```
TRACE_PAGES_TI   = 2
TRACE_PAGES_GC = 0 (OLTP) or 20 (LVC)
TRACE_PAGES_AL  = 5
TRACE_PAGES_DW = 3
TRACE_PAGES_US  = 10
TRACE_PAGES_UT = 5
TRACE_PAGES_SV = 5
TRACE_PAGES_EV _= 2
TRACE_PAGES_CS  = 2
TRACE_PAGES_BUP = 0 (task type does not exist, yet)
```

All tasks in a UKT pack their trace pages in a single location, from which each task can obtain pages (with trace entries). Therefore, in a single user case a task can use all pages in the UKT. The trace pages of the other UKTs cannot be used.
The kernel trace size is not just information for an external file; it also describes memory used in the main memory during the runtime of the database kernel.

2.4 xserver.prt

The file `xserver.prt` is in the directory `<independent_data_path>/wrk`. Any errors that occur during X Server communication are entered in the file `xserver.prt`. See also section 7.5.

2.5 core

On UNIX, a `core` is often written after crashes; this file contains the stack back trace. This is very important for troubleshooting.
Only a complete `core` is useful for troubleshooting. Since the `core` is written to the run directory of the database, this directory must be large enough to hold the entire `core`.
The size of the `core` depends on the configured caches, since their content is included in the information written to the `core`.
As of Version 7.4.02 Build 04 the default parameter SUPRESS_CORE=YES specifies that the core is not written, since a stack back trace is logged in `knldiag`.

2.6 DrWtsn32.log

On Windows, the stack back trace is logged in the file `DrWtsn32.log`. This file is usually located in the directory `C:\winnt` (Microsoft Windows NT) or `C:\Documents and Settings\All Users\Documents\DrWatson` (Microsoft Windows 2000).

2.7 appldiag

UNIX Systems:

The file `appldiag` is stored in the directory `$DBROOT/wrk/<user_name>` (Version 6.2) or in the directory `<independent_data_path>`[1]`/wrk/<user_name>` (as of Version 7.2). A separate file is written for each user (<sid>adm, sqd<sid> and root).

Windows Systems:

The file `appldiag` is written only if the environment variable DIAGFILE is set to **`YES`**. It is stored directly under `%DBROOT%\wrk` (Version 6.2) or `<independent_data_path>/wrk` (as of Version 7.2).

This file logs the error messages of the runtime environment for the database.

Example:

```
02-26 15:56:49 15480 ERR 11546 XPARAM   Could not find xparam file
02-26 15:56:49 15480 ERR 11545 XPARAM   Could not open xparam file:
'/sapdb/data/config/S11'
03-26 11:15:25 14961 ERR -11608 COMMUNIC sql03_request: wrong
connection state, state is 'requested'
```

2.8 knldump

An emergency shutdown writes the global memory to the file `knldump`. The file system must be large enough for this. Development teams can use X_DIAGNOSE to analyze `knldump`, if necessary.

3 Different Trace Types

Trace Type	Activation	Usage
Database Trace (Vtrace)	Database Manager (GUI/CLI,)	Analysis of database kernel problems, including SQL errors and performance problems
Precompiler Trace	Environment variable SQLOPT	Analysis of SQL errors and interface problems between the application and the database kernel

3.1 Database Trace (Vtrace)

The database kernel can write a database trace (Vtrace). Developers can use the trace to see which chunks of code have been run. You can use options to specify certain modules for tracing.

[1] See section 5.1

The database trace can have a negative effect on the performance of the system (Versions older than 7.2 and liveCache), so activate it for analysis purposes only.
As of Version 7.2, you can activate the database trace in the OLTP environment without any significant effects on performance. After reproducing the problem, you must deactivate and flush the trace.
Flushing writes the information from the cache to the file `knltrace`. If you forget to flush the trace, some (versions older than 7.x) or all information will be missing from the trace, since the trace is initially only written to the main memory. The information is not written to disk until the trace is flushed.
You can activate the database trace only if the database has (at least) the operational state COLD (or ADMIN). You cannot activate the database trace in the state OFFLINE.
Version 7.2:

- You can activate the database trace (Vtrace) with the Database Manager GUI: Choose *Check → Kernel Trace*. Here, you can also select whether you just want to activate the DEFAULT database trace , or other traces as well. You can also activate the trace for a certain session (choose *Advanced → Trace Session*, and enter the session ID), and set it so that it is deactivated if a certain error occurs (choose *Advance → Stop on Error*, and enter the error number).
  After reproducing the problem, you must deactivate and flush the database trace. (Choose the button with the red jagged line. DO NOT choose the button with the red cross; this deletes the database trace.) Choose *Protocol* to convert the file `knltrace` into a readable file.

- You can also activate the database trace with the Database Manager CLI:
  `dbmcli -d <database_name> -u <dbm_user>,<password> -uUTL -c util_execute diagnose vtrace default on`
  Deactivate the database trace with the following command:
  `dbmcli -d <database_name> -u <dbm_user>,<password> -uUTL -c util_execute diagnose vtrace default off`
  Flush the database trace with the following commands:

  - Database has state COLD (ADMIN):
    `dbmcli -d <database_name> -u <dbm_user>,<password> -uUTL -c util_execute diagnose vtrace`
  - Database has state WARM (ONLINE):
    `dbmcli -d <database_name> -u <dbm_user>,<password> -uSQL -c sql_execute vtrace`

  Analyze the database trace with the following command:
  `dbmcli -d <database_name> -n <server_name> trace_prot abknx`

  The Database Manager CLI command `trace_prot` generates the file `<database_name>.prt` in the run directory.

As of SAP DB Version 7.x, an active trace no longer cause performance bottlenecks in the OLTP environment. This means that there is nothing to stop you activating the database trace by default.

However, we recommend that you do not activate the database trace constantly in the liveCache environment; use it only in error situations.

Example of a database trace:

```
************************************************************************
***     KERNPROT  7.2.5                      2002-11-08  11:32:24     ***
************************************************************************

Input File: G:\sapdb\data\wrk\HKN\knltrace.dat

---------- from entry 1.6 to 100.8197 (from page 1 to 100) -------------

===== T5 ===== id4115722 ==================================1.6  page 1
RECEIVE: ascii, full_swap, 70205-DBM     (1 segment, len: 40)
(1.6  page 1)
   ok / RETURN SEGMENT 1   (0 parts, len: 40)
         sqlstate: '00000'

REQUEST: ascii, full_swap, 70205-DBM     (1 segment, len: 80)
(1.102  page 1)
   dbs SEGMENT 1   (1 part, len: 80)
         session_sqlmode, user_cmd
   command PART   (1 argument, size: 32232)
         buf(19): 'COMMIT WORK RELEASE'
```

3.2 Precompiler Trace

The precompiler is the interface between the application and the database. A trace from this interface contains the SQL statements sent from the application to the database kernel, their parameters, and the results.

Example of a Precompiler Trace:

```
PRODUCT : liveCache C/C++ Precompiler Runtime
VERSION : 7.1.4
BUILD   : 032-000-055-840

version  :P_1, P_2
SQL STATEMENT  :  FROM MODULE : dbslada            AT LINE : 4186
OUTPUT :  LZU  :  NT/INTEL  7.1.4    Build 032-000-055-840
OUTPUT :  PCR  :  C-PreComp 7.1.4    Build 032-000-055-840
START  :  DATE :  2001-07-13    TIME :  0013:01:01
END    :  DATE :  2001-07-13    TIME :  0013:01:01
SESSION   : 1;
SQLMODE   : SAPR3   AT DATABASE :  DB_000
SERVERDB  : S10
SERVERNODE:
OPTION-CONNECT :
CONNECT "SAPR3             " IDENTIFIED BY :A  SQLMODE SAPR3  ISOLATION LEVEL 0
 TIMEOUT 0
SQL STATEMENT  :  FROM MODULE : dbslada            AT LINE : 6390
START  :  DATE :  2001-07-13    TIME :  0013:01:01
END    :  DATE :  2001-07-13    TIME :  0013:01:01
```

- **Precompiler Versions as of 7.2.04:**
  You can use program IRTRACE to activate the precompiler trace.

  This uses a shared memory segment to communicate with interface runtime. If irtrace is used to make changes to an interface runtime trace, then a corresponding entry is

made in the shared memory segment. Interface runtime checks the entries in the shared memory regularly, and makes corresponding changes to its trace. An assignment is made with the process ID. As long as the corresponding process is active, the entry remains in the shared memory, from where you can query it. Before you can use a shared memory segment, you must create a synchronization file, which the processes use to access the shared memory. The file `irtrace.shm` is stored (for all versions) in the directory (`<independent_data_path>/wrk`). The subdirectory `wrk` must exist. Read and write authorizations must also be given to IRTRACE and the interface runtime.

The tool gives you the following options for changing the precompiler trace:

- Activating/deactivating/switching the trace for a process:
  `irtrace -p <process_id> -t <trace_type>`
  You can choose from the following trace types:
  - `long :` long trace
  - `short :` short trace
  - `off :` deactivate trace
- Activating/deactivating the trace for all interface processes on the application server:
  `irtrace -p all -t <trace_type>`
- Displaying the current trace settings of an interface runtime:
  `irtrace -p <process_id>`
- Displaying an overview of the trace settings of all interface runtimes where changes have been made with IRTRACE:
  `irtrace -p all`
- If a faulty installation, for example, means that IRTRACE cannot find the shared memory file, then you can specify the path explicitly (`-f <path>`). However, this path applies only to IRTRACE; the interface runtime continues to find the shared memory file from the environment.
- Display entire shared memory contents (for support purposes):
  `irtrace -s`

- **All Database Versions**

  You can use the environment variable SQLOPT to activate a precompiler trace for the currently used shell (UNIX) or the opened command prompt (Windows).

  Activate:

  - Set the variable SQLOPT to the value **`-X -F <file_name>.pct`**. This writes `*.pct` files in the current directory, which then contain information on the source of errors.
  - You can also use the option **`-X -F PID`**, which includes the process ID in the file name.

  Deactivate:

- Reset the variable SQLOPT.

- **Other Precompiler Log Files**
  As of Precompiler Version 7.3.00, the file `ldrdiag.pct` is written to the directory `<independent_data_path>` (run directory). If this is not possible, the file is written to the work directory of the precompiler application. If this is also not possible, the file is written to stderr.

  This file contains information on:

  - the access path
  - the version of the application
  - the version of the precompiler runtime used by the application
  - Any errors that occurred when the runtime was loaded

  Example:

```
2002-08-05 16:25:28 0xa28 OK  Application .\lobtst002 7.4.3 started.
2002-08-05 16:25:28 0xa28 ERR Library not found:
D:/SAPDevelop/V74/develop/usr/runtime/7403\pgm\libpcr.
2002-08-05 16:26:12 0xa28 OK  Application .\lobtst002 7.4.3 started.
2002-08-05 16:26:12 0xa28 OK
D:/SAPDevelop/V74/develop/usr/runtime/7403\pgm\libpcr 7.4.3 load
succeed.
```

  Each application that is started in the same directory writes to the same `ldrdiag.pct` file. The file can grow to a size of 50KB. When it reaches this size, the file is renamed as `ldrdiag.pct.old` and a new `lrddiag.pct` file is created.

  You can use the following command to activate and deactivate the trace for the precompiler runtime load in `ldrdiag.pct`.

```
irconf -l [on|off]
```

  The trace is deactivated by default.

  Use the command

```
irconf -l
```

  to check whether the trace is activated or deactivated.

4 Error Numbers and Messages and Their Meaning

4.1 Database Error Messages

4.2 Database Errors -9xxx

The following chapters describe errors with the numbers -9xxx.

4.2.1 Overview of the –9xxx Errors and Their Categories

Error Number	Category	Page
-9001	1	**Error! Bookmark not defined.**
-9002	1	**Error! Bookmark not defined.**
-9003	1	**Error! Bookmark not defined.**
-9004	2	35
-9005	1	**Error! Bookmark not defined.**
-9006	1	**Error! Bookmark not defined.**
-9007	2	35
-9008	1	**Error! Bookmark not define**

Error Number	Category	Page
-9026	1	**Error! Bookmark not defined.**
-9027	3	36
-9028	1	**Error! Bookmark not defined.**
-9029	1	**Error! Bookmark not defined.**
-9030	4	**Error! Bookmark not defined.**
-9033	3	36
-9034	3	36
-9041	1	**Error! Bookmark not define**

Error Number	Category	Page
		d.
-9010	1	Error! Bookmark not defined.
-9013	1	Error! Bookmark not defined.
-9014	1	Error! Bookmark not defined.
-9018	1	Error! Bookmark not defined.
-9020	3	36
-9021	1	Error! Bookmark not defined.
-9023	1	Error! Bookmark not defined.
-9024	1	Error! Bookmark not

Error Number	Category	Page
		d.
-9044	1	Error! Bookmark not defined.
-9045	1	Error! Bookmark not defined.
-9046	3	36
-9050	2	35
-9051	2	35
-9052	2	35
-9053	1	Error! Bookmark not defined.
-9111	2	35

Error Number	Category	Page
		defined.
-9025	1	**Error! Bookmark not defined.**
Others	Others	37

Error Number	Category	Page
-9209	4	**Error! Bookmark not defined.**

4.2.2 Error Category 1

Category 1 errors refer to either basis tables or indexes.
In almost all cases, the error is caused by a hardware error. Always check the hardware (disk or controller) if you encounter an error of this type.

Number	Error Message	Analysis and Solution
-9001	e_invalid_root	**For index: Section 4.2.2, 4. Step, paragraph I** **For table: Section 4.2.2, 4. Step, paragraph III**
-9002	e_illegal_branchlength	
-9003	e_illegal_entrylength	
-9005	e_illegal_key	
-9006	e_illegal_keylength	
-9008	e_illegal_record	
-9010	e_invalid_invlistpos	
-9013	e_invalid_index_structure	
-9014	e_invalid_leaves_structure	
-9018	e_page_in_wrong_tree	
-9021	e_no_statistic	
-9023	e_illegal_entrypos	

Number	Error Message	Analysis and Solution
-9024	e_invalid_entrypos	
-9025	e_illegal_invlistpos	
-9026	e_bad_datapage	**For index: Section 4.2.2, 4. Step, paragraph I** **For table: Section 4.2.2, 4. Step, paragraph II**
-9028	e_bad_file	
-9029	e_bad_invfile	**For index: Section 4.2.2, 4. Step, paragraph I** **For table: Section 4.2.2, 4. Step, paragraph III**
-9041	e_file_not_accessible	
-9044	e_inconsistent_nodetype	
-9045	e_root_check	
-9053	e_data_page_corrupt	

Analysis:

Before you analyze the error, you must back up all files from the run directory (see section 2 on version dependencies), since you may need to examine them.
Use the analysis tools and files to find out whether the object with the error is an index or a table.

1. **Step: Is there a reference to a root page number in knldiag/knldiag.err?**

   This reference could appear as follows:

   Example:

```
   10-09 16:56:28    16 ERR 53132 DATACACH Illeg PNO 3951285
PTR 00AEA70000000000
   10-09 16:56:28    16 ERR 53000 B*TREE
00000000000000000000000000000000
   10-09 16:56:28    16 ERR 53000 B*TREE   ROOT  978858
```

   - YES, a similar entry appears

     Continue with step 2

   - NO ↯

     Contact a SAP DB expert.

2. **Step: Is the database in the state WARM (ONLINE)?**

Use the command `x_cons <database_name> show state` or the command `dbmcli -d <database_name> -u <dbm_user>,<password> db_state` to find out the operational state of the database.

- Yes the database is WARM (ONLINE)

Continue with step 3

- No the database is OFFLINE or COLD (ADMIN)

Start the database
Check the state of the database again (repeat step 2)

  - The database could not be started.

Back up all diagnosis files again (see section 2 on version dependencies)
Check `Knldiag` for errors
Analyze and remove any errors
Start the database and continue with step 3 when the database is WARM (ONLINE)

  - ↯ The error could not be identified or removed when the database was started, and the database is still OFFLINE or COLD (ADMIN).

Contact a SAP DB expert.

3. Step: What type of object is it?

Use the root page number and the following SELECT statement (using SQL Studio or Database Manager CLI) to access the ROOTS table:

Example:
SELECT * FROM roots WHERE root = 978858

Result:

```
TABLEID          ¦ OWNER ¦ TABLENAME ¦ INDEXNAME  ¦ TYPE        ¦     ROOT
-----------------+-------+-----------+------------+-------------+----------
00000000000010F5 ¦ SAPR3 ¦ TCESYSTT  ¦ TCESYSTT~1 ¦ NAMED INDEX ¦   978858
                 ¦       ¦           ¦            ¦             ¦
```

The TYPE column determines the type of the database object. The solution to the problem depends on the type of the object with the error.

4. Step: Solution

Type : Named Index	Solution under I
Type : Table	Solution under II and III
Type : Short String Column	Solution under IV

I. Solution for Type NAMED INDEX

Since the object with the error is an index, and this object can be reconstructed with the data from the basis table, this problem can be solved by the following action:

The name of the index is in the INDEXNAME column.
The name of the table in which the index was created is in the TABLENAME column.

Determine index structure with SQL Studio:

```
SELECT * FROM sysodbcindexes WHERE table_name =
'<table_name>'
```

Example:

```
SELECT * FROM sysodbcindexes WHERE table_name = 'MSEG'
```

Result:

```
INDEX_NAME | TYPE | SEQ_IN_INDEX | COLUMN_NAME
-----------+------+--------------+------------
MSEG~M     |  3 |         1 | MANDT
MSEG~M     |  3 |         2 | MATNR
MSEG~M     |  3 |         3 | WERKS
MSEG~M     |  3 |         4 | LGORT
MSEG~M     |  3 |         5 | BWART
MSEG~M     |  3 |         6 | SOBKZ
MSEG~R     |  3 |         1 | MANDT
MSEG~R     |  3 |         2 | RSNUM
MSEG~S     |  3 |         1 | SMBLN
MSEG~S     |  3 |         2 | SJAHR
MSEG~S     |  3 |         3 | SMBLP
```

Make a note of this output, since you will need the index structure later when you create the index again.

Delete and create the index again with SQL Studio.

Delete the index with SQL Studio:

Command:
```
DROP INDEX <index_name> ON <table_name>
```
Example:
```
DROP INDEX "MSEG~R" ON mseg
```

Create the index:

Command:
```
CREATE INDEX <index_name> ON <table_name>
(<column>, <column>.....)
```
Example:
```
CREATE INDEX "MSEG~R" ON mseg (mandt, rsnum)
```

After deleting and creating the index again, proceed with step 5.

II. Solution for Type Table, Special Case -9026 and -9028:

If one of these two errors occurs in a table, then you can try to remove the error with the statement `CHECK TABLE`. You can execute CHECK TABLE with SQL Studio.

Command: `CHECK TABLE <table_name>`
Example: `CHECK TABLE mseg`

- CHECK TABLE runs without errors → proceed with step 5
- CHECK TABLE terminates again with an error → proceed with III

III. Solution for Type Table for All Other Errors in Category 1

If the object with the error is a table, then the data in the table has been destroyed. The solution to the problem depends on whether it is one of two cases.

a) The table contains data that can be generated from other tables
b) The table contains data that cannot be generated from other tables

You must contact the responsible application developer as quickly as possible to decide whether the table is case a) or case b).

- If the data can be generated, then proceed as instructed by the developer. Then continue with step 5.
- If you cannot generate the data, then you must recover the entire database.
- If you cannot decide one way or another, then you must recover the database.

IV. Solution for Type Short String Column

If the column with the error is the short column of a blob[2], then the data of the blob has been destroyed. The solution to the problem depends on whether it is one of two cases.

a) The table to which the BLOB belongs contains data that can be generated.
b) The table to which the BLOB belongs contains data that cannot be generated.

You must contact the responsible application developer as quickly as possible to decide whether the table is case a) or case b).

- If the data can be generated, then proceed as instructed by the developer. Then continue with step 5.
- If you cannot generate the data, then you must recover the entire database.
- If you cannot decide one way or another, then you must recover the database.

5. Step: Follow-Up Actions

To make sure that no other problems occur after you remove the error, you must start a check database structure (Verify) in the system. You can use the tool CONTROL

[2] Binary Large Object

(*Backup* → *Save* → *Verify Devspaces*) or Database Manager GUI (*Check* → *Database*).

For details about check database structure, see section **Error! Reference source not found.**.

4.2.3 Error Category 2

All category 2 errors are problems that might not reoccur after restarting the database.

Number	Error Message	Analysis and Solution
-9004	e_illegal_filename	**Can the error be reproduced after restarting the database?** **➔ Generate database trace.** **➔ Contact a SAP DB expert.**
-9007	e_illegal_page_no	
-9050	e_illegal_page_address	
-9051	e_invalid_fbm_mark	
-9052	e_multiple_converter_entry	
-9111	e_move_error	

Analysis:

Before you analyze the error, you must back up all files in the run directory (see section 2 on version dependencies), since you may need to examine them.

1. **Step: Which action produced the error?**

   Your first step is to determine the cause of the problem. You need to know the trigger so that you can later determine whether your solution was successful.

2. **Step: Stop and Start Database**

   - Stop the database instance.
   - Start the database instance.
     - If you could start the database instance successfully, proceed with step 3.
     - If you cannot reach the operational state WARM (ONLINE), contact a SAP DB expert.

3. **Step: Reproducing the Problem**

   Attempt to reproduce the problem, using the information gathered about the cause of the problem in step 1.

   - **No** – the error could not be reproduced
     ➔ Problem solved

- **Yes - error still occurs**

Reproduce the problem with activated database trace
Contact a SAP DB expert.

4.2.4 Error Category 3

Category 3 errors require you to recover the database.

Number	Error Message	Analysis and Solution
-9020	e_init_missing	**Recovery**
-9027	e_bad_fdir	
-9033	e_bad_syspage	
-9034	e_bad_usmpage	
-9046	e_no_converter_entry	

4.2.5 Error Category 4

Category 4 errors usually indicate an inconsistency in the log devspace.

Number	Error Message	Analysis and Solution
-9030	e_bad_logpage	**Restore devspace**
-9209	e_log_error	

Analysis:

1. **Step: Which log mode is configured?**

   You can find out the log configuration as follows:

   Version 7.2:
   Database Manager GUI: Choose *Information* → *Log*
   Database Manager CLI*:*

```
dbmcli -d <database_name> -u <dbm_user>,<password>
param_directget LOG_MODE
```

2. **Step: Removing the Error**

   a) SINGLE log mode:
   SINGLE log mode does not mirror a log at database level. No restore devspaces (volumes) are possible in this configuration.

Using RAID:
If a customer uses a raid system for mirroring a log area, then you must use RAID administration functions to remove any defects in the log.

No log device mirroring:
Note: This case should not occur in production systems.
In this case, you can recover the database with existing backups only. Any data that is not backed up is lost. You must recover the database with *Instance Install*, so that the log devspaces (volumes) are reinitialized.

b) DUAL log mode:
The DUAL log mode lets you use a second (mirrored) log area to copy the defective log devspace (volume). You do not need to recover the database. Version 7.2 requires you to use the Database Manager GUI for this.

4.2.6 Other -9xxx Error

If you encounter errors with the number-9xxx that cannot be assigned to any of the preceding categories, then you must contact a SAP DB expert.

4.3 Error Messages in the File knldiag

This section describes error messages that can be found in the file `knldiag` or `knldiag.err`. Unfortunately, this list cannot describe all possible errors.

4.3.1 Error Messages that Can Occur in All Instance Types

```
FBM      invalid devno(26) or offset(1015382)
```

This error message specifies that, when the database is restarted, the converter requests that the FBM sets the block address 26/1015382 (block 1015382 on device 26) to 'occupied'. However, there are only 20 devspaces (volumes) on this example, and the FBM rejects the request. This error can be caused, for example, by a defect on the converter side; this defect means that at least the logical DeviceNo has switched to 26.

```
singleio write_error! NO ERROR(0)#
singleio devno is 7#
I/O      #
I/O      LOGDEV DEVNO: 1#
I/O      /sapdb/PLC/log/DISKL001#
vmarkbad pos 262135 marked as bad#
SHUTDOWN continued_log/disk_not_accessibl
vfwrite   EOF after 7168 bytes#
```

Less data than expected was written to the devspace (volume) (devno 7). If the device is a raw device, then the limit of the raw device has been reached (for example, because a size has been specified incorrectly). From the system's perspective, the write action has been successful, and no error is registered (NO ERROR). If this error occurs, then you must check the size of the devspace (volume).

```
03-15 11:57:19    13 WNG 53409 INDEX    Realloc Of Dynamic Vector Not
Possible
03-15 11:57:22    15     53412 INDEX    All Perm Leaf Pages Read: 325
03-15 11:57:22    15     53412 INDEX    All Perm Records Read: 35506
03-15 11:57:22    15     53412 INDEX    Start Final Merge Step
```

```
03-15 11:57:23    15     53412 INDEX    Stop Final Merge Step: 1
```

These messages indicate that an internal structure is not large enough to include all the data from a parallel CREATE INDEX. For this reason, the index is set up in sequence. The size of this structure is estimated at the start of CREATE INDEX, and can not be increased afterwards. This structure is usually large enough, but not always.

```
B*TREE   BD53: predsep.k > sep.k: 0
SHUTDOWN column/invalid_leaves_st
SHUTDOWN partial_rollback/shutdown
SHUTDOWN *** EMERGENCY ***: 9140
```

SAP DB stores all data from tables and indexes in B* trees. The B* trees are sorted in ascending order according to the key sequence.
This error occurs when the key (and separator) sequence is not sorted in ascending order in the B* tree. This is usually caused by a hardware problem.

4.4 Operating System Return Codes

If messages similar to the following are logged in `knldiag`, then *rc* indicates a return code from the operating system:

```
06-21 10:40:04      0x198 ERR 18008 TASKING  Could not create thread:
'ASYNCi', rc = 8
07-24 15:04:36      0x273 ERR 18491 IO       file/tape/pipe write
error, rc = 112
09-09 14:32:02      0x120 ERR 19206 MESSAGES Could not write to event
log, rc = 1717
09-12 15:58:18       0xCE ERR 19207 MESSAGES Event log is full, rc =
1502
```

You can find out what these return codes mean with the tools SYSRC and SYSTEMRC (Windows only) or XSYSRC. In the Versions 7.2.05 older than Build 25 and 7.3.00 older than Build 31, the tools SYSRC and SYSTEMRC are located in the directory `<dependent_path>/pgm`. Since this directory is not in the path, you must specify a complete path to call these tools.
As of Versions 7.2.05 Build 25, 7.3.00 Build 31, 7.4.02 Build 16, 7.4.03 Build 10, 7.4.04 Build 00, the tool XSYSRC is shipped in `<independent_data_path>/bin`, which means you can call it without specifying the path.

Call:

```
xsysrc <rc>
<dependent_path>/pgm/sysrc <rc>
<dependent_path>\pgm\systemrc <rc>
```

Example:

```
C:\SAPDB\LCA\db\pgm>sysrc.exe 8
Errortext for errorcode 8:
'Not enough storage is available to process this command.'

C:\ \SAPDB\LCA\db\pgm>sysrc.exe 112
Errortext for errorcode 112:
'There is not enough space on the disk.'
```

4.5 UNIX Signals

To find out which signal a number is (for example, signal 6), you can execute the command `kill -l` on the relevant operating system.

Example on SUN:

```
$ kill -l
```

```
HUP INT QUIT ILL TRAP ABRT EMT FPE KILL BUS SEGV
SYS PIPE ALRM TERM USR1 USR2 CLD PWR WINCH URG
POLL STOP TSTP CONT TTIN TTOU VTALRM PROF XCPU XFSZ
WAITING LWP FREEZE THAW CANCEL LOST RTMIN RTMIN+1
RTMIN+2 RTMIN+3 RTMAX-3 RTMAX-2 RTMAX-1 RTMAX
```

In this case, you have to count which signal has which number.

Example on AIX:

```
$ kill -l
 1) HUP       14) ALRM      27) MSG       40) bad trap  53) bad trap
 2) INT       15) TERM      28) WINCH     41) bad trap  54) bad trap
 3) QUIT      16) URG       29) PWR       42) bad trap  55) bad trap
 4) ILL       17) STOP      30) USR1      43) bad trap  56) bad trap
 5) TRAP      18) TSTP      31) USR2      44) bad trap  57) bad trap
 6) ABRT      19) CONT      32) PROF      45) bad trap  58) bad trap
 7) EMT       20) CHLD      33) DANGER    46) bad trap  59) CPUFAIL
 8) FPE       21) TTIN      34) VTALRM    47) bad trap  60) GRANT
 9) KILL      22) TTOU      35) MIGRATE   48) bad trap  61) RETRACT
10) BUS       23) IO        36) PRE       49) bad trap  62) SOUND
11) SEGV      24) XCPU      37) bad trap  50) bad trap  63) SAK
12) SYS       25) XFSZ      38) bad trap  51) bad trap
13) PIPE      26) bad trap  39) bad trap  52) bad trap
```

5 Directory Structure

For documentation about the directory structure, see the documentation *The SAP DB Database System → Directory Structure of the Database System → SAP DB Directories*.

5.1 DBROOT-Free Installation

As of Version 7.2.04 Build 04, the environment variable **DBROOT** no longer exists. This new structure without DBROOT means that it is possible to operate multiple databases with different versions on the same host. This is, however, not recommended in production system environments for performance reasons (Exception: UNIX 64 bit).

Explanation of the new terms:

- **<dependent_path> (INSTROOT):**
  The version-dependent database software is stored in this directory. You can use the following command to determine where this directory is stored on customers' hosts:
  `dbmcli -d <database_name> db_enum`

  Example:
  `dbmcli -d E20 db_enum`

  Result:

```
OK
E20     /sapdb/E20/db   7.2.4.0 fast    running
```

```
E20     /sapdb/E20/db   7.2.4.0 slow    offline
```

On this UNIX system, the directory is stored under `/sapdb/E20/db`.

- **<independent_data_path> (IndepDataPath):**

  The version-independent data (including, for example, the run directory) is stored in the directory defined with `<independent_data_path>` . You can use the following command to determine which directory the customer has defined with **<independent_data_path>**:

```
dbmcli –d <database_name> -u <dbm_user>,<password>
dbm_getpath INDEPDATAPATH
```

  Example:

```
dbmcli –d E20 –u control,control dbm_getpath
INDEPDATAPATH
```

  Result:

```
OK
/sapdb/data
```

On this UNIX system, the `<independent_data_path>` directory is stored under `/sapdb/data`.

- **<independent_program_path> (IndepProgPath):**

  The version-independent database programs are stored in the directory defined with `<independent_program_path>`. You can use the following command to determine which directory the customer has defined with **<independent_program_path>**:

```
dbmcli –d <database_name> -u <dbm_user>,<password>
dbm_getpath INDEPPROGPATH
```

  Example:

```
dbmcli –d E20 –u control,control dbm_getpath
INDEPPROGPATH
```

  Result:

```
OK
/sapdb/programs
```

On this UNIX system, the `<independent_program_path>` directory is stored under `/sapdb/programs`.

How must the path variable be set?

As of Version 7.2.04 Build 15:

Microsoft Windows: `<independent_program_path>\bin` and `<independent_program_path>\pgm`

UNIX: `<independent_program_path>/bin`

The variable **DBROOT** is not set.

6 What Happens when the Database Is Started?

6.1 Explanation of the Individual Steps with knldiag Examples

Most of the examples in the following text are taken from a version 7.3.00 database. The procedure is similar in versions 7.2 and 7.4, although the sequence of the entries in `knldiag` can vary to some extent. If an example is taken from a different version, this is explicitly indicated.

1) First, the system reads the parameters (at the very beginning, the run directory and the TASKCLUSTER are of particular interest. The `knldiag` file is then created in the run directory).
   If the last stop of the database was not performed using a shutdown, the system writes the DIAGHISTORY. To do this, the system first copies `knldiag` to `knldiag.old` and opens a new `knldiag`, which already contains a log of the fact that the files are backed up.

   Example:

```
2002-11-18 13:17:03      0x734     19841 DIAGHIST Backup of diagnostic files is in
progress
2002-11-18 13:17:05      0x734     19842 DIAGHIST Backup of diagnostic files has
finished
```

   Problem: If a problem occurs during the reading of the parameters, this cannot yet be logged in `knldiag`. In this case, you receive only the error message `kernel died before reaching cold state`. Under Microsoft Windows, there is often additional information in the Event Log in this case. If the `x_start` tool is still available, you can use this to start the kernel. Unlike `dbmsrv`, this tool does not suppress the message output to STDOUT . Alternatively, you can also start the program `kernel` directly with the database name as a parameter to see the error message. However, if the operational state COLD (ADMIN) is then reached, the shell in which the command was called is terminated. As of version 7.3.00 build 29 and 7.4.03 Build 05, `knldiag` is opened before the parameters are read. In this case, problems that occur can be logged directly in `knldiag`.

2) The TASKCLUSTER is logged in the `knldiag` file. The number of the various tasks here do not indicate how many of each type of task are actually created, but rather how many of each type can run in a User Kernel Thread (UKT) .
   The number of tasks of each type that are actually created is also logged in `knldiag`. BUP: Backup Task (not yet implemented; for future use); DW: Pager (Data Writer); US: User Task; SV: Server Task; EV: Event Task.

   Example:

```
2002-11-18 13:17:06      0x734     19692 TCLUSTER
tw;al;ut;2000*sv,100*bup;10*ev,10*gc;ti,100*dw,cs;30000*us;compress
2002-11-18 13:17:06      0x734     19693 TCLUSTER number of 'BUP':   0
2002-11-18 13:17:06      0x734     19693 TCLUSTER number of ' DW':   8
2002-11-18 13:17:06      0x734     19693 TCLUSTER number of ' US':  10
2002-11-18 13:17:06      0x734     19693 TCLUSTER number of ' SV':  12
2002-11-18 13:17:06      0x734     19693 TCLUSTER number of ' EV':   0
2002-11-18 13:17:06      0x734     19693 TCLUSTER number of ' GC':   0
```

3) Eventing is used only rarely. It is possible to use parameters to set that events that occur (such as DB full, error occurred, ...) are logged in the file `knldiag.evt` and placed in the event queue. From there, events can then be retrieved using event tasks and the Database Manager GUI could display corresponding events. However, in most cases, `knldiag` will indicate that output to `knldiag.evt` is suppressed and that no event buffer was created.

   Example:

```
2002-11-18 13:17:06      0x734     19811 EVENTING Suppressing output to 'knldiag.evt'
0 pages
2002-11-18 13:17:06      0x734     19813 EVENTING No Eventbuffer created (0 entries
100 tasks)
```

4) After this, some information about the database to be started is logged in `knldiag`. This includes, among other information, the database name, the host name, the process ID, the number of processors, and information about the available memory.

   Example:

```
2002-11-18 13:17:06   0x734     19769 INFO  Starting SERVERDB:              'DB73'
2002-11-18 13:17:06   0x734     19770 INFO           SERVERNODE:           'P59960'
2002-11-18 13:17:06   0x734     19771 INFO           Process ID:              1700
2002-11-18 13:17:06   0x734     19773 INFO           Date:                 02-11-18
2002-11-18 13:17:06   0x734     19772 INFO           Owner:                'SYSTEM'
2002-11-18 13:17:06   0x734     19775 INFO           Number of Processors:       1
2002-11-18 13:17:06   0x734     19782 INFO           Fiber:                   'NO'
2002-11-18 13:17:06   0x734     19776 INFO           Max virtual memory:   2047 MB
2002-11-18 13:17:06   0x734     19777 INFO           Total physical memory: 255 MB
2002-11-18 13:17:06   0x734     19778 INFO         Available physical memory: 7 MB
2002-11-18 13:17:06   0x734     19779 INFO         Kernel shared data size:  28 MB
2002-11-18 13:17:06   0x734 WRN 19425 DBSTATE  Shared exceeds available memory
```

   In this example, the system also outputs the warning that there is not enough physical memory available and that the server will therefore probably page.

5) The memory for SHAREDDYNDATA and SHAREDDYNPOOL and for the converter cache is then requested. SHAREDDYNDATA and SHAREDDYNPOOL are two memory pools from which various memory requests are later fulfilled. In the case of SHAREDDYNDATA, this is memory space suitable for I/O, which is requested by pages. In the case of SHAREDDYNPOOL, this is an area from which the memory can be requested by bytes. The data cache is not counted as part of SHAREDDYNDATA.
   Requests for memory for the SHAREDDYNPOOL are not recorded in `knldiag`, while both other types of memory request are logged.

   Example:

```
2002-11-18 13:17:06      0x734     19837 MEMORY   235 Pages allocated for
SHAREDDYNDATA at 0x013C0000
2002-11-18 13:17:06      0x734     19837 MEMORY   6 Pages allocated for CONVERTER
CACHE at 0x015A0000
```

6) The status of the COM trace is then output before the various threads are started. The actions up to now were all performed by the coordinator thread. Therefore, the same TID appears before all of the messages in `knldiag`.
   The Thread ID that has been assigned to each thread is displayed in front of that thread. In addition to the standard threads, at least six user kernel threads (UKTs) are

always started (for Trace Writer TW, Log Writer AL, Utility Task UT, Server Tasks SV, Timer Task TI, Pager (Data Writer) DW, and for User Tasks US).

Example:

```
2002-11-18 13:17:06      0x734     19831 TRACE_IO Configured _MAX_MESSAGE_FILES:0
2002-11-18 13:17:06      0x734     19832 TRACE_IO Disabled: No TraceMessagefiles
wanted
2002-11-18 13:17:06      0x74C     19688 THREAD   CLOCK thread started
2002-11-18 13:17:06      0x6DC     19688 THREAD   TIMER thread started
2002-11-18 13:17:06      0x748     19688 THREAD   ASYNC0 thread started
2002-11-18 13:17:06      0x744     19688 THREAD   DCOM0 thread started
2002-11-18 13:17:06      0x6B4     19688 THREAD   CONSOLE thread started
2002-11-18 13:17:06      0x76C     19688 THREAD   REQUESTOR thread started
2002-11-18 13:17:06      0x508     19627 TASKING  UKT started, TID:0x508
2002-11-18 13:17:06      0x228     19627 TASKING  UKT started, TID:0x228
2002-11-18 13:17:06      0x628     19627 TASKING  UKT started, TID:0x628
2002-11-18 13:17:06      0x134     19627 TASKING  UKT started, TID:0x134
2002-11-18 13:17:06      0x7AC     19627 TASKING  UKT started, TID:0x7AC
2002-11-18 13:17:06      0x75C     19627 TASKING  UKT started, TID:0x75C
```

7) The UKT, which contains the user tasks (same TID for the messages in `knldiag`), then attaches the `knltrace` and the associated I/O thread is started.

Example:

```
2002-11-18 13:17:06      0x75C     19615 DEVIO    Attaching devspace 'knltrace'
2002-11-18 13:17:06      0x758     19613 DBSTATE  I/O thread for 'knltrace' started
```

8) The memory requests from the SHAREDDYNPOOL and SHAREDDYNDATA areas are then performed. These are all identified with the *dynpool* or *dynDATA* and multiple lines may be connected. For example, the system might first display the number of pagers (data writers), and then how many bytes are required for the task description of each pager, and how many bytes are required for the cache list. The number of pagers multiplied with the total of the byte specifications then gives the value that is output for DYNP_B12_DATA_WRIT.
However, it may also be that the total number of bytes is specified first, and then the breakdown (as, for example, in the case of DYNP_B15_DEV_DESC).

Example:

```
2002-11-18 13:17:06      0x75C     54003 dynpool  NUM DATAWRITER          :      8
2002-11-18 13:17:06      0x75C     54003 dynpool  DATAWRITER task desc    :     12
2002-11-18 13:17:06      0x75C     54003 dynpool  DATAWRITER cache list   :      8
2002-11-18 13:17:06      0x75C     54003 dynpool  DYNP_B12_DATA_WRIT      :    160
2002-11-18 13:17:06      0x75C     54003 dynDATA  DYND_B15_CONFIG         :      3
2002-11-18 13:17:06      0x75C     54003 dynpool  DYNP_B15_DEV_DESC       :    120
2002-11-18 13:17:06      0x75C     54003 dynpool  PHYS_SYS_DEV_DESCR      :     24
2002-11-18 13:17:06      0x75C     54003 dynpool  PHYS_DATA_DEV_DESCR     :     48
2002-11-18 13:17:06      0x75C     54003 dynpool  PHYS_LOG_DEV_DESCR      :     48
...
```

9) The version specifications for the kernel and the runtime environment are displayed among the memory requests.

Example:

```
2002-11-18 13:17:06      0x75C     19600 VERSION  'Kernel    7.3.0    Build 029-000-
087-809'
2002-11-18 13:17:06      0x75C     19600 VERSION  'NT/INTEL  7.3.0    Build 029-000-
087-809'
```

10) Following the memory requests, the system starts the pagers (data writers) (at least one for each data cache region – if the parameter MAXDATADEVSPACES is set to a value greater than _DATA_CACHE_RGNS, as many data writers are started, as devspaces (volumes) are possible. Then multiple data writers can access the same critical region).

Example:

```
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 13
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 0
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 12
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 1
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 11
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 2
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 10
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 3
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 9
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 4
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 8
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 5
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 7
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 6
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT start pid: 6
2002-11-18 13:17:06      0x75C     53040 I/O      DATAWRIT first datacache: 7
```

11) If the message `DBSTATE SERVERDB is ready` is entered in `knldiag`, the database is in the COLD (ADMIN) operational state and all user tasks and the utility task are in the "Connect Wait" state.

Example:

```
2002-11-18 13:17:06      0x6DC     19601 DBSTATE  SERVERDB is ready
```

12) For starting from COLD (ADMIN) to WARM (ONLINE), the system first performs a connect of the utility task. The Database Manager performs various actions at this point, if it is appropriately configured - such as activating the database trace (Vtrace) or the events. These actions are performed using the utility task and are encapsulated; that is, for each action, a connect to the utility task is performed and then released again.

Example:

```
2002-11-18 13:17:07      0x628     19633 CONNECT  Connect req. (T5, Node:'',
PID:1972)
2002-11-18 13:17:07      0x628     19651 CONNECT  Connection released, T5
2002-11-18 13:17:07      0x628     19633 CONNECT  Connect req. (T5, Node:'',
PID:1972)
```

13) The devspaces (Volumes) are then attached in the configured sequence and the corresponding I/O threads are generated, that is, first the system devspace (not in 7.4, as there is no longer a system devspace in this version and the converter is stored on the data volumes), then DATADEV_0001, then DATADEV_0002, and so on, and finally, the log devspaces (Volumes).
After the system devspace (or the first data volume in version 7.4) has been opened, the restart record is imported.
When the devspaces (volumes) are opened, the system also performs a few checks. These include the runtime environment checking its "magic page". In versions 7.2 and 7.3, the total of the individual data devspace sizes (volume sizes) is compared with the value from the restart record. If these values do not match, the restart is

terminated. In this way, you can avoid the database allowing itself to be started if the correct devspaces (volumes) are not available. However, if the devspaces (volumes) are switched for other devspaces (volumes) of the same size, the system will not notice this.
This size entry in the restart record exists especially for this comparison and no longer exists in version 7.4. The reason for this is that in version 7.4, there is an IO Man Info Page (I/O Manager) on each volume, in which the following information is stored:

- Capacity
- Logical ID
- Previous logical ID
- Next logical ID
- Bad Flag (specifies whether the volume can no longer be opened due to an error)
- ResetBadFlagCount (Specifies how often the bad flag has already been reset using a command)
- db_ident (from Init Config timepoint)

You can use this data to determine more accurately whether the correct volumes have been attached. You can also determine whether the volumes are still in the original sequence using the chaining of the volumes using previous and next logical ID. This is important, as the position of the data pages (pages) is stored in the converter in the format “Page xyz auf DEVNO n, Offset abc”. If two volumes have been switched, DEVNO n refers to an entirely different volume, and the data can no longer be found.
The IO Man Info Page must be newly created during the migration from version 7.3 to 7.4. This is performed during the inplace migration.
Log volumes also contain an additional Log Info Page, which contains, among other information, the current db_ident.
If no error occurs, you see no indication of any of these checks in `knldiag`. Normally, the system outputs only the messages with regard to the attaching and the starting of the I/O thread.

Example:

```
2002-11-18 13:17:07      0x628     19615 DEVIO    Attaching devspace 'SYS_001'
2002-11-18 13:17:07      0x760     19613 DBSTATE  I/O thread for 'SYS_001' started
2002-11-18 13:17:07      0x628     19617 DEVIO    Single I/O attach, 'SYS_001', UKT:3
2002-11-18 13:17:07      0x628     19615 DEVIO    Attaching devspace 'DAT_0001'
2002-11-18 13:17:07      0x61C     19613 DBSTATE  I/O thread for 'DAT_0001' started
2002-11-18 13:17:07      0x628     19615 DEVIO    Attaching devspace 'LOG_001'
2002-11-18 13:17:07      0x704     19613 DBSTATE  I/O thread for 'LOG_001' started
```

14) The converter cache is then created, and the converter is imported. In versions below 7.4.03, there are no messages relating to this in `knldiag`, unless any errors occur.

Example - Version 7.4.03:

```
2002-11-20 10:25:40      0x874        12 Pager    Start Read Converter
2002-11-20 10:25:40      0x874        13 Pager    Stop Read Converter, Pages: 9 IO: 9
```

15) The system then requests additional memory from SHAREDDYNPOOL and SHAREDDYNDATA .

Example:

```
2002-11-18 13:17:07      0x628     54003 dynpool  PNO_SUPPLY_SIZE        :  10000
2002-11-18 13:17:07      0x628     54003 dynpool  PNO_SUPPLY elem        :      4
2002-11-18 13:17:07      0x628     54003 dynpool  B10_PNO_POOL_PIDQ      :    608
2002-11-18 13:17:07      0x628     54003 dynpool  US + SV + DW +  8      :     38
2002-11-18 13:17:07      0x628     54003 dynpool  B10_PNO_POOL_PIDQ elem :     16
2002-11-18 13:17:07      0x628     54003 dynpool  DYNP_B10_PNO_POOL      :  40608
...
```

16) The system then creates the data cache. When doing this, the system requests the number of memory blocks specified with the parameter _DATA_CACHE_RGNS . The size of the individual critical regions corresponds to DATA_CACHE/_DATA_CACHE_RGNS. The system rounds (down) during the calculation, meaning that it does not request exactly the amount of memory that is specified with DATA_CACHE. In this example, eight critical regions, each of 370 pages are created; that is, of 40 pages less than specified.

Example:

```
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x032B0000
2002-11-18 13:17:07      0x628     54003 dynDATA  DATA_CACHE_PAGES       :    370
2002-11-18 13:17:07      0x628     54003 dynpool  B20_DATACACHE_CB       :  25160
2002-11-18 13:17:07      0x628     54003 dynpool  DATACACHE num cblocks  :    370
2002-11-18 13:17:07      0x628     54003 dynpool  DATACACHE cblock size  :     68
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x035A0000
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x03890000
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x03B80000
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x03E70000
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x04160000
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x04450000
2002-11-18 13:17:07      0x628     19837 MEMORY   370 Pages allocated for DATA CACHE
at 0x04740000
```

17) After the system has created the data cache, it imports the file directory. When the system does this, it checks the structure of the tree, but more extensive checks are not performed. There are also no messages for this in `knldiag`, unless errors occur. Possible errors are: "`No converter entry`", "`Bad data page`", and "`Invalid leaf structure`".

18) In version 7.4, the number of history files for a liveCache instance is also checked. The system logs how many history files are found and how many files can exist. If a backup has been imported, or the parameter MAXUSERTASKS has been changed, unnecessary history files might exist. If this is the case, the unnecessary history files are immediately deleted together with their history.
In the following example, ten history files were found and there should also be ten in accordance with the current configuration.

Example -Version 7.4.03:

```
2002-11-20 10:25:41      0x874        20 Log      History: 10 (10) files existing
2002-11-20 10:25:41      0x874        23 Log      History: all history files
registered, GC is ready
2002-11-20 10:25:41      0x584     19617 VOLUMEIO Single I/O attach, 'DAT_0001',
UKT:5
```

19) The system analyzes the log area. To do this, the system reads the Log Info Page (with Single I/O using the utility task, therefore, there is a message “Single I/O attach” in `knldiag` before this - the specified TID is that of the UKT that contains the utility task).
The Log Info Page is written to the offset 0 (Version 7.4) or 2 (Version 7.3) of the log devspaces (volume) at regular intervals. This gives an indication of the appearance of the log data written to the log area, to which data is written cyclically. If the Log Info Page records that the log has been deleted, the system does not perform any further checks on the log area. Otherwise, the system reads the first page (the page after the Log Info Page) and the last log page (last offset). The system then searches for certain points on the log volume that must be found, as otherwise the log will not match the data. On the Log Info Page, there is always an offset for a logical ID (in Version 7.3 LPNO, in Version 7.4 IO sequence), on which it must be possible to find this page.
The system first searches for the last known page (LastKnown). If there are more log entries after this, the system reads to the end (in version 7.3, in blocks, that is _MULT_IO_COUNT pages are read together, in version 7.4, sequentially), to find the current log end. After the last page, the system then searches for the first (oldest) page (FirstKnown). This is usually found immediately after the last known page.
To be safe, the system searches for the position of the first page that has not been saved (FirstNotSaved).
After the system has found the last savepoint from the restart record, which the system must redo from the log, in the case of a restart, the system compares the DB identifier from the restart record with that of the Log Info Page. Only if all checks were successful can the system continue with the rest of the restart.

When the system performs redos based on the log, the first and last log page numbers are logged in `knldiag`, together with REDO messages. You can use these messages to see how many log pages have already been redone.

Example - Version 7.3:

```
2002-11-18 13:17:07      0x628     19617 DEVIO    Single I/O attach, 'LOG_001', UKT:3
2002-11-18 13:17:07      0x628     52680 RESTART  first log page 2990
2002-11-18 13:17:07      0x628     52680 RESTART  last  log page 3937
2002-11-18 13:17:12      0x628     52680 RESTART  redo  log page 3802
```

In version 7.4, the system outputs additional messages about logging.
These include, for example, the number of configured log queues and the flush mode used. In the following example, this is “MinimizeSpace” and means that the system writes to an offset more than once.
“OldestNotSaved” is displayed with the IO sequence and the offset, as are the oldest page in the log (FirstKnown), and the first and last log position, which are required for the restart.
“last redo read” provides brief information about the last redone log entry. In the following example, it is the 4514th redone entry, a COMMIT of transaction 4782 and its 1009th REDO-entry. This entry was written with the IO sequence 6836 on the volume offset 972 and on the byte offset 8116 in the log page. This was written on 20.11.2002 at 10:25:21.

In version 7.4, the reasons for the writing of a savepoint are written in `knldiag`. In this case, you can see that the savepoint manager was requested to write a closing savepoint for the restart .

Example - Version 7.4.03:

```
2002-11-20 10:25:41      0x874         7 Log      1 queues, flushmode is
'MinimizeSpace', devstate is 'Okay'
2002-11-20 10:25:41      0x874         8 Log      Oldest not saved is ioseq 3703 @
off 1595
2002-11-20 10:25:41      0x874         9 Log      First known on LogVolume is ioseq
3703 @ off 1595
2002-11-20 10:25:41      0x874         6 Log      Restart from ioseq 6452 @ off 855
to ioseq 6836 @ off 972
2002-11-20 10:25:41      0x874        10 Log      Result after scanning the log
device: 'Ok'
2002-11-20 10:25:41      0x774        13 Log      recovering log from log_volume from
IOSeq: '6452'
2002-11-20 10:25:43      0x774        15 Log      ArchivLog: normal end at off 972
lastseq 6836.
2002-11-20 10:25:43      0x774        18 Log      last-redo-
read#4514:TR4782(1009)[6836]@972.8116'UpdateRecord':20021120:102521
2002-11-20 10:25:43      0x874        17 Log      Savepoint requested by T4 reason
'Restart' (started).
```

20) Once all log information has been redone, the system writes a checkpoint (Version 7.4: savepoint).

Example:

```
2002-11-18 13:17:12      0x134     53040 SAVPOINT (1) Start Write Data: 0
2002-11-18 13:17:12      0x134     53040 SAVPOINT (1) Stop Data I/O Pages 227 IO 69
2002-11-18 13:17:12      0x134     53040 SAVPOINT (2) Start Prepare Data T: 33
2002-11-18 13:17:12      0x134     53070 SAVPOINT B20PREPARE_SVP: 9
2002-11-18 13:17:12      0x134     53040 SAVPOINT (2) Stop Data I/O Pages 0
2002-11-18 13:17:12      0x134     53040 SAVPOINT (3) Start Write Data: 0
2002-11-18 13:17:12      0x134     53040 SAVPOINT (3) Stop Data I/O Pages 0
2002-11-18 13:17:12      0x134     53040 SAVPOINT (3) Start Write PSE36: 0
2002-11-18 13:17:12      0x134     53040 SAVPOINT (3) Stop PSE36 I/O Pages 0
2002-11-18 13:17:12      0x134     53040 SAVPOINT (3) Start Write Conv: 0
2002-11-18 13:17:12      0x134     53040 SAVPOINT (3) Stop Conv I/O Pages 2 IO 1
2002-11-18 13:17:12      0x134     53071 SAVPOINT B20SVP_COMPLETED: 9
2002-11-18 13:17:12      0x134     53040 CHKPOINT (2) Start Prepare Data T: 33
2002-11-18 13:17:12      0x134     53070 CHKPOINT B20PREPARE_SVP: 10
2002-11-18 13:17:12      0x134     53040 CHKPOINT (2) Stop Data I/O Pages 0
2002-11-18 13:17:12      0x134     53040 CHKPOINT (3) Start Write Data: 0
2002-11-18 13:17:12      0x134     53040 CHKPOINT (3) Stop Data I/O Pages 1 IO 1
2002-11-18 13:17:12      0x134     53040 CHKPOINT (3) Start Write PSE36: 0
2002-11-18 13:17:12      0x134     53040 CHKPOINT (3) Stop PSE36 I/O Pages 0
2002-11-18 13:17:12      0x134     53040 CHKPOINT (3) Start Write Conv: 0
2002-11-18 13:17:12      0x134     53040 CHKPOINT (3) Stop Conv I/O Pages 1 IO 1
2002-11-18 13:17:12      0x228     19617 DEVIO    Single I/O attach, 'LOG_001', UKT:2
2002-11-18 13:17:12      0x134     53071 CHKPOINT B20SVP_COMPLETED: 10
```

21) If you are starting a version >= 7.4.02 build 05 for the first time, the system performs a migration of the undo log and redo log files. This is necessary, as the root ID in these files may not be correct in versions <= 7.4.02 build 03. The following message then appears in the `knldiag` file: "`History: waiting for removal of complete history`".
The system then waits for the garbage collectors delete all history pages. This can take a long time, as the history was not deleted promptly in older versions. During this action, the utility task is in the "vsleep" state.

22) If the database was started successfully, a message is logged that the database is in the WARM (ONLINE) operational state.

Example:

```
2002-11-18 13:17:12      0x628     52612 RESTART  LOCAL: ready
2002-11-18 13:17:12      0x628     54003 dynpool  DYNP_A42_PARSEID       :    120
2002-11-18 13:17:12      0x628     54003 dynpool  UNICODE MBYTE_TOTALSIZE:      0
2002-11-18 13:17:12      0x628     19602 DBSTATE  SERVERDB is in WARM mode
```

23) If the AUTOSAVE LOG was activated before the last shutdown, the Database Manageractivates it again. This command is sent to the kernel using the utility task, meaning that another connect is logged for the utility task.

Example:

```
2002-11-18 13:17:12      0x628     19633 CONNECT  Connect req. (T5, Node:'',
PID:1972)
2002-11-18 13:17:12      0x628     52104 AUTOSAVE standby mode on
2002-11-18 13:17:12      0x628     19651 CONNECT  Connection released, T5
```

6.2 Additional Information

6.2.1 What do the Letter and Number Combinations for the Memory Requests Mean?

Each area is uniquely identified by a combination. You can see the respective names in the table below:

Name	Meaning
AK51 / A51	Catalog Cache
K38	Backup
KB50	Lock list
K51 / LOCK	Lock list elements
K55	Log Queue
K56	Rollback cache
K57	Restart record
KB90 / K90	Server tasks
B10	Free block management (not in 7.4)
B12	Pager (Data writer) (not in 7.4)
B15	I/O manager (not in 7.4)
B16	Converter (not in 7.4)
B20	Data cache
B77 / TREELOCK	Treelock list
B94	History
B930	Garbage collector

6.2.2 Difference Between “Attach” and “Single I/O Attach”

The “normal” attach is performed in preparation for multi-user operation. If multiple tasks are active in a thread, the I/O requests are placed in the I/O queue and processed using the started I/O thread. The relevant task remains inactive until the I/O request has been processed.
Under Microsoft Windows, asynchronous I/O is usually performed, if single I/O is not performed. This is quicker, since a system call is saved. The I/O request is transferred to a system call, which responds immediately. This means that the task is not blocked.

Instead, the task receives an “I/O pending” message and makes itself inactive. An asynchronous I/O thread (the worker thread) determines when an asynchronous I/O is completed and wakes the corresponding task.
The I/O thread mechanism is only used under Microsoft Windows if the number of concurrently running I/O requests is exhausted.
Under UNIX, this asynchronous I/O is connected with a lot of overhead, meaning that it is not used.
A single I/O is performed if only one task is active in the UKT. An independent file handle is created for this, so that two I/O requests can read at two different positions on the disk , without having to perform another seek (when writing/reading to sequential blocks). There is a single I/O handle for each devspace (Volume) for each UKT, which is created when the first single I/O is performed (→ Single I/O Attach).

6.2.3 When Does the DB Identifier Change?

The db_ident changes if an action is performed that deletes log entries. This means that the log history is interrupted, and that a complete backup must be created again. The actions concerned are:

- Init Config (Instance Install) with Restore
- Restart Until
- Restore Until
- Changing the log mode to DEMO
- CLEAR LOG
- SET LOG VOLUME OFF (7.4 only)

6.2.4 Term Explanation: LPNO, IO Sequence, Offset

In Versions below 7.4, every time a new log page was written, a new ID was assigned to it(LPNO) and the corresponding page was written to a new position on the disk (Offset). As there is a fixed number of writable positions, you can use the LPNO to calculate the offset, if you know the offset of the very first LPNO ((LPNO MOD Number)+1.Position).
As of Version 7.4, you can set the log flush mode using a database parameter. With the option “Minimize Space” a page can be written more than once to the same position on the disk, if it was not full the first time it was written. A new ID is assigned for every write operation (IO Sequence). As you do not know how many IO sequences were written to an offset (and the number can also be different for each offset), you can no longer calculate the offset from the IO sequence. Therefore, the information about a log page always consists of the IO sequence and the offset. If you are dealing with a specific log entry, the offset is also still specified in the page:
IOSeq@Offset(Volume).Offset(Page)
If the flush mode “Maximize Safety” is used (not yet implemented), a page is written to a new position with every write operation. This means that log pages may not be completely full, but it ensures that a log page cannot break down during overwriting. With this procedure, you could also calculate the offset from the IO sequence.

Version 7.4 (IOSequence)	Log Info	0 - 3 q+1	... ...	p - p+2	p+3	…	q

Offset		0	1	...	m	m+1	…	o
IOManager View	IO Man Info	1	2	...	m+1	IO Man Info	1	...	n
	Log Volume 1					Log Volume 2		

In Version 7.4, there is an IO Man Info Page on every volume. This is not visible for logging. All log volumes are viewed by the logging as one area - the offset is continuously numbered. The cyclical log writing begins only after the log info page – that is, at offset 1.

Version 7.2 (LPNO)	Conf 1	Conf 2	Log Info	0 r+1	1 r+2	2	…				r
Offset	0	1	2	3	4	5	…	m	m+1	...	m+n+1
BD View	0	1	2	3	4	5	…	m	0	…	n
	Log Devspace 1								Log Devspace 2		

In Versions before 7.4, there is no IOManInfo Page. However, all log devspaces are still viewed as one area with continuously numbered offsets by the logging . There are two config pages and the log info page. Only after this - that is, at offset 3 - does the cyclical log writing begin.

7 Instance Type-Independent Problems

7.1 Database Is Full

Symptom:

If the database is full (no more space available on the devspaces (Volumes)), this becomes apparent to the user by the system “hanging”. All tasks are suspended, until space is available again.

Analysis:

There are a number of ways to check whether the database is full:

1. Database Manager GUI: Double click the database symbol:
   You can see, in the bar display the fill level of the data devspaces (Volumes).
2. Database Manager GUI: Choose *Check* → *Server* → *active*:
   If the database is full, the active tasks are in the ‘db_full’ state.
3. Database console:
   The console can be called at operating system level with the command
   `x_cons <database_name> show active`
4. `knldiag`:
   Before the database is full, messages are already logged in the file `knldiag` that indicate that the database will soon be full:
   ```
   2001-08-14 09:26:04      0x7EC WRN 53025 DATA     PERM  USED 99(1 PAGES LEFT)
   ```
   These messages are logged as of a fill level of 80%. If the database is then

actually full, the following message is written in this file for every suspended task:

```
2001-08-14 09:26:04      0x7EC WRN 53000 DATA     DB FULL task suspended: 15
```

Solution:

If the database is full, a new data devspace (Volume) must be added. You can do this using Database Manager GUI.

7.2 Log Full

Symptom:

If the log area is full (no more space available on the log devspaces(Volumes)), this becomes apparent to the user by the system "hanging". All tasks are suspended, until space is available again.

Analysis:

There are a number of ways to check whether the log area is full:

1. Database Manager GUI: Double click the database symbol.
   You can see, in the bar display the fill level of the log devspaces (Volumes).
2. Database Manager GUI: Choose *Check → Server → active*
   If the log area is full, the active tasks are in the 'Vsuspend' state. The number in brackets after the status refers to the reason for the suspend. You can determine this by choosing *Check → Server → Suspends*.
3. Database console:
   You can call the console at operating system level with the command
   `x_cons <database_name> show active`
4. `knldiag`:
   Before the log area is actually full, messages are already logged in the file `knldiag` that indicate that the log area will soon be full:
   ```
   2001-08-14 09:16:27      0x7EC WRN 52438 LOG      log used 50(499 pages left)
   ```
   These messages are logged as of a log fill level of 50%. If the log area then actually becomes full, the following message is written in this file for every suspended task:
   ```
   2001-08-14 09:16:36      0x7EC WRN 52437 LOG      LOGFULL/suspended pid:15
   ```

Solution:

If the log area is full, you must create a log backup. You can do this using the Database Manager GUI . After the log area has been backed up, the old log information can be overwritten. The suspended database tasks continue with their work when the first log segment has been backed up. It is not necessary to wait until the entire log area has been backed up.

Preventative Measures:

To avoid the log area becoming full and therefore that you must constantly monitor the log fill level, the log area can be automatically backed up. To do this, in Database Manager GUI, choose the option *Backup → Autolog On/Off*. If this option is activated, the log information is backed up as soon as a log segment is full . The size of the log segments is determined using the database parameter LOG_SEGMENT_SIZE . The

value 0 for this parameter means that a third of the entire log size is used for the size of a log segment.
You can also use the Database Manager GUI to check that automatic log backup is really activated. In the detail display of the database status (double click the database symbol), you can see the “Autolog” line with the appropriate status.

7.3 Database Crash

The following files must be saved for analysis before the first restart attempt after the crash (see section 2 with regard to version dependencies):
`knldiag`, `knldiag.err`, `knltrace`, `knldump`, `rtedump`, `dbm.*` files, `c<pid>.*` files (AK-DUMP, up to version 7.2.04) or `ak.*` files (AK-DUMP, from version 7.2.05).

You can find the stack backtrace for the crash in `core` (UNIX) or in `DrWtsn32.log` (Microsoft Windows). If no `DrWtsn32.log` was written, you can also determine the function in which the error occurred as follows:

- If the crash was caused by an exception, there is a message similar to the following in the `knldiag` or `knldiag.err` file:

  ```
  EXCEPT EXCEPTION:0xc0000005 Addr:0x57fe7d ( 0:0x1529cd4c:0:0 )
  ```

- You can find the mapfiles for the relevant database version in the directory `%DBROOT%\support\mapfiles` (Version 6.2) or `<dependent_path>\support\mapfiles` (Version 7.2). The crash address is usually a function from the `kernel.map` file:

  ```
  kernel

   Timestamp is 3829a5c3 (Wed Nov 10 18:05:07 1999)

   Preferred load address is 00400000

   Start         Length     Name                   Class
   0001:00000000 00342810H .text                  CODE
   0002:00000000 00000139H CODE32                 CODE
   0003:00000000 00001378H .rdata                 DATA
   0003:00001378 00000000H .edata                 DATA
   0004:00000000 00000004H .CRT$XCA               DATA
   0004:00000004 00000004H .CRT$XCZ               DATA
   0004:00000008 00000004H .CRT$XIA               DATA
   0004:0000000c 00000008H .CRT$XIC               DATA
   0004:00000014 00000004H .CRT$XIZ               DATA
   0004:00000018 00000004H .CRT$XPA               DATA
   0004:0000001c 00000004H .CRT$XPX               DATA
   0004:00000020 00000004H .CRT$XPZ               DATA
   0004:00000024 00000004H .CRT$XTA               DATA
   0004:00000028 00000004H .CRT$XTZ               DATA
   0004:00000030 000507f9H .data                  DATA
   0004:00050830 000315a0H .bss                   DATA
   0005:00000000 00000064H .idata$2               DATA
   0005:00000064 00000014H .idata$3               DATA
   0005:00000078 00000318H .idata$4               DATA
   0005:00000390 00000318H .idata$5               DATA
   0005:000006a8 00000db4H .idata$6               DATA
   0006:00000000 00000400H .rsrc$01               DATA
   0006:00000400 000019a4H .rsrc$02               DATA

    Address         Publics by Value              Rva+Base   Lib:Object

   0001:00000000       _sql80k_NewSrvState        00412000 f vos80kc.o
  ```

```
 0001:00000130       _sql80k_CtrlHandler@4      00412130 f vos80kc.o
 0001:00000200       _sql80k_ServiceMain@8      00412200 f vos80kc.o
 0001:00000350       _WinMain@16                00412350 f vos80kc.o
 0001:000004b0       _sql80kn_main              004124b0 f vos80knc.o
 0001:000005b0       _sql80kn_write_msg         004125b0 f vos80knc.o
 0001:000006f0       _sql80kn_set_db_state      004126f0 f vos80knc.o
 0001:000007b0       _sql80kn_dlg_proc@16       004127b0 f vos80knc.o
 0001:00001820       _ak91set_copyright         00413820 f vak91.o
 0001:00001880       _ak91connect               00413880 f vak91.o
 0001:00001a80       _ak91free_mem              00413a80 f vak91.o
 0001:00001b40       _ak91init_acv              00413b40 f vak91.o
...
 0001:0016b340       _a720prepare_two_qual      0057d340 f ak4lib:vak720.o
 0001:0016bf10       _a720qual_eval_check       0057df10 f ak4lib:vak720.o
 0001:0016c060       _a720trace_eval            0057e060 f ak4lib:vak720.o
 0001:0016c180       _a720glob_init_eval_stat   0057e180 f ak4lib:vak720.o
 0001:0016c360       _ak70all_ands              0057e360 f ak4lib:vak70.o
 0001:0016c8b0       _ak70analyze_condition     0057e8b0 f ak4lib:vak70.o
 0001:0016d2c0       _ak70build_allo            0057f2c0 f ak4lib:vak70.o
 0001:0016d610       _ak70check_level           0057f610 f ak4lib:vak70.o
 0001:0016db80       _ak70one_and               0057fb80 f ak4lib:vak70.o
 0001:0016e100       _ak70or_qual               00580100 f ak4lib:vak70.o
 0001:0016e3f0       _ak70order_multi           005803f0 f ak4lib:vak70.o
 0001:0016ed10       _ak70_test_order_multi     00580d10 f ak4lib:vak70.o
 0001:0016f120       _ak70qual_evaluation_rec   00581120 f ak4lib:vak70.o
 0001:0016f730       _ak70skip_stack_entries    00581730 f ak4lib:vak70.o
 0001:0016f8a0       _ak70more_index_strat      005818a0 f ak4lib:vak70.o
 0001:00170160       _ak70only_index_strat      00582160 f ak4lib:vak70.o
 0001:001726d0       _ak70expl_inv              005846d0 f ak4lib:vak70.o
...
```

- You can search for the address in this file. You should note, however, that it is seldom possible to find the exact address. The address in the mapfile is the start address of a function, however the crash occurs in the middle of a function.
  You should always search for the first characters of the address. In this example, you should search for **57f**. The address is found in the column **after** the function name. The function for which you are searching is the one with the next smaller address relative to the crash address - in this example, the function **_ak70one_and**.
- You must always search for the relevant database patches (exact version including build number) in the mapfile. The function for which you are searching could be at a different address in another patch.

7.4 Error Analysis if the System “Hangs”

7.4.1 UNIX

The procedure described in the following applies for UNIX systems except Reliant and Linux. On these two platforms, there is a (Clone) process for every thread (irrespective of whether it is a UKT or an I/O thread). As these cannot be assigned to the “responsible“ database thread, the procedure cannot be used for these platforms.

The database may be in a state in which you can no longer see what is currently happening and everything is blocked. If you have already collected all of the other information (such as active Tasks, memory usage, x_cons show all, ...) and cannot contact a developer, it can be useful to generate a callstack and restart the database. This callstack can then be evaluated by the developers later to find the cause of the error.
To do this, you must first determine the correct process ID. To do this, run a ps command:

```
ps –afe | grep kernel
```

Example:

```
ds8803:s11adm 3> ps -afe | grep kernel
root          0      0  0.0   May 24 ??          36:18.79 [kernel idle]
sqds11    14625      1  0.0   May 27 ??           2:35.32
/sapdb/S11/db/pgm/kernel S11
sqds11    14749  14625  0.0   May 27 ??          14:57:07
/sapdb/S11/db/pgm/kernel S11
s11adm     2997   2956  0.0 14:33:16 ttyp0        0:00.01 grep kernel
```

TWO database kernel processes are output. One of the processes is the parent of the other. This parent has the init process as its parent (PPID 1). In this example, process 14625 (second column) has the init process (third column) as its parent. Process 14749 has process 14625 as its parent.
The child process must be terminated by sending signal 10 twice:

```
kill -10 <database_pid>
kill -10 <database_pid>
```

Example:

```
kill -10 14749
kill -10 14749
```

A `core` that contains the callstack is then written.

CAUTION:

Before you terminate the database process, you should check the following:

1. NO checkpoint is written at this termination.
2. Is there enough space in the file system to record the `core`? Is the file system set up in such a way that files larger than 2 GB can be created?
3. Are the ulimits set in such a way that the `core` can be created completely?
4. If the memory allocated by the kernel (Data Cache) is very large (multiple GB), writing the `core` will take a very long time (it may take several hours - at one customer, 80 GB were written in two hours – and this was with very fast disks). The database cannot be restarted during this time. Can the customer accept a downtime as long as this?

7.4.2 Microsoft Windows (NT and 2000)

Under Microsoft Windows (NT and 2000), you can also create a stack backtrace. To do this, you must determine the process ID of the *kernel* process from the file `knldiag`. You can then run the command `Drwtsn32 -p <pid>`. You can find the backtrace in the file `Drwtsn32.log`.

7.5 Error Analysis if Problems Occur with the X Server

Note: All commands shown here are created on the basis of 7.4.
The X Serverallows remote SQL communication.
The command

```
x_server -V
```

outputs the version of the X Server.

The command

```
x_server -h
```

displays the possible options (Install, Remove, Start, and Stop) of the X Server that can be performed by customers.

Installing the X Server
Before the X Server can be started, it must be registered once. You can do this with the command

```
x_server install
```

There is always only one X Server on a host that performs the communication for all database instances on this host. The newest X server must always be installed.

Deleting the X Server Registration
The command

```
x_server remove
```

can be used to delete the registration again.

Starting the X Server
The command

```
x_server start
```

starts the X Server; that is, the Vserver process. Remote communication is then possible. If Remote SQL is activated (*x_server start*) , that is, accesses to the database over a network are allowed, the Vserver process is generated. A new Vserver process is generated for every user process that logs on to the database remotely. The generating process serves this user process, and the new Vserver process waits for the next user logon. Under Microsoft Windows NT/2000, the X Server runs as a service.

Errors when starting the X Server are not logged in `xserver.prt` .
For example the following message might be displayed on the screen if errors occur when starting the X Server:

```
18655 ERROR: Wrong XSERVER version installed! Please reinstall the XSERVER.
   Installed:   'W32/INTEL 7.4.2    Build 003-000-000-000'
   Expected:    'W32/INTEL 7.4.3    Build 000-000-000-000'
```

If this error occurs, the newest X Server is not installed or registered on this host. In this case, the X Server must be uninstalled and the newest version installed (registered).

Command Sequence**:**

```
x_server remove; x_server install; x_server start
```

Stopping the X Server
The command

```
x_server stop
```

deactivates remote communication and therefore ends all X Server processes. The command

`x_server -k`

can be used to end all X Server processes in the same way as the option stop.

Log File of the X Server

If errors occur during communication through the X Server, these are entered in the file `xserver.prt` (UNIX: < 7.3.00 Build 29 `xserver.prot`).
The file `xserver.prt` is in the directory `<independent_data_path>/wrk`.
The file `xserver.prt` is newly created after the X Server has been successfully started. At every restart of the X Server, the file `xserver.prt` is copied to `xserver.old`. The X Server log is cyclically rewritten under UNIX and Microsoft Windows as of 7.4.
The size of the X Server log is defined when the X Server is started. By default, the file `xserver.prt` is 64 k in size. Under Microsoft Windows, the size is stored in the registry.
In special analysis cases, you can use the command

`x_server -Z <size_in_KB>`

to change the size of `xserver.prt`.

Structure of xserver.prt

Messages of the X Server

```
-------------------------------------------------------------------------------------------
-----
|Date.      |Time     |TID(hex)    |Type| MsgID|Label    |Message text
|-------------------------------------------------------------------------------------------
-----|2002-11-01 |06:50:28 |    0xEE    |    | 19707|CONNECT  |WINSOCK: 2.0, SQLTCP-DLL:
7.4.3                                    |
|2002-11-01 |06:50:28 |    0xEE    |    | 19751|XSERVER  |XSERVER started, 'W32/INTEL
7.4.3         |-------------------------------------------------------------------------------
-------------------
|2002-11-08 |10:37:41 |   0x149    |ERR | 18393|CONNECT  |Socket receive error, rc = 10054
|
|2002-11-08 |10:37:41 |  0x149   |ERR | 18393|CONNECT  |Socket receive error, rc = 329                 |
|                                                          receive error [10054]
|2002-11-08 |10:37:41 |   0x149    |    | 19840|XSERVER  |'P67043' disconnect, Reference:
329
```

Every message in `xserver.prt` receives a timestamp (columns date and time).
The **TID(hex)** column specifies the process ID (UNIX) or the thread ID (Microsoft Windows) of the relevant X Server.
MSGID specifies the number of the error message - unfortunately, these error messages are not contained in the system tables.
LABEL specifies the software component. This makes it easier to identify the area in which the message occurred.
Message text specifies an explanatory error message text. You can use the return code (rc= or [errno]) specified here to determine the cause of the error. The program SYSRC (UNIX) or SYSTEMRC (Microsoft Windows) is available (newer versions are called XSYSRC, see section 4.4) with which the error numbers can be decoded.
For example: rc=10054 stands for *Unknown Identifier (rc = 10054)*
An existing connection was forcibly closed by the remote host

Debug Options of the X Server
The command

```
x_server -D<debug_level>
```

starts the X Server in Debug Mode. More detailed messages are then written to the file `xserver.prt`. Different debug levels are supported. If an X Server is running, you can use the command `x_server -N <debug_level>` to change the debug level. The debug level should only be activated by a SAP DB expert to analyze X Server problems. The communication between client and X Server is logged here: Connects, ports used, and so on.
The debug level should only be used during the analysis, as it can have a negative influence on performance, depending on the debug level selected.
Debug levels 1-9 are supported. The higher the debug level, the more detailed the output is (1 is the lowest debug level, and 9 is the highest debug level).

Additional Options

x_server –Y	UNIX only, prevents the NI SERVER processes being started

Options that Are Used Only by Developers

x_server –S<service_port_number>	Select an alternative port
x_server –X	Suppresses the port that is used for 6.1

7.6 ILLEGAL DATA DEV SIZES Error

If the ILLEGAL DATA DEV SIZES occurs, the value `crMaxDataPno` +1 from the restart record does not match the total of the devspace sizes in the parameter file. You must then investigate why a different size of the database is stored in the restart record from the size that is actually configured.

Example:

```
2002-12-30 11:28:27      0x648     19615 DEVIO    Attaching devspace
'E:\sapdb\KE1\dbsys\SYS'
2002-12-30 11:28:27      0x638     19613 DBSTATE  I/O thread for
'E:\sapdb\KE1\dbsys\SYS' started
2002-12-30 11:28:27      0x648     19617 DEVIO    Single I/O attach,
'E:\sapdb\KE1\dbsys\SYS', UKT:3
2002-12-30 11:28:27      0x648     19615 DEVIO    Attaching devspace
'E:\sapdb\KE1\sapdata\DISKD0001'
2002-12-30 11:28:27      0x67C     19613 DBSTATE  I/O thread for
'E:\sapdb\KE1\sapdata\DISKD0001' started
2002-12-30 11:28:28      0x648     19615 DEVIO    Attaching devspace
'E:\sapdb\KE1\sapdata\DISKD0002'
2002-12-30 11:28:28      0x600     19613 DBSTATE  I/O thread for
'E:\sapdb\KE1\sapdata\DISKD0002' started
2002-12-30 11:28:28      0x648     19615 DEVIO    Attaching devspace
'h:\sapdb\KE1\saplog\DISKL001'
2002-12-30 11:28:28      0x16C     19613 DBSTATE  I/O thread for
'h:\sapdb\KE1\saplog\DISKL001' started
2002-12-30 11:28:28      0x648 ERR 53007 CONFIG   ILLEGAL DATA DEV
SIZES
```

```
2002-12-30 11:28:28      0x648 ERR 18196 DBCRASH  vabort:Emergency
Shutdown, vbd15.c: 5137
```

7.7 Problem with the Log Page Numbers

A Log Page Overflow can occur due to using SAP DB versions for years (maximum value of log count: 2**31).
Use the following SQL statement to determine whether your log page numbering has reached critical values:

- SAP DB Versions 7.2, 7.3
  `SELECT value FROM internal_state WHERE description = 'forced last known page no'`
- SAP DB Versions 7.4
  `SELECT value FROM internal_state WHERE description = 'last known sequence'`

If your log page numbering has reached critical value (value is 200,000,000 less than 2**31), you can reset numbering using CLEAR LOG command from Database Manager (GUI or CLI). Note that the backup history is discontinued as a result and you must start a new history by performing a new full backup.

1. Shut down the database instance
   `dbmcli –d <database_name> -u <dbm_user>,<password> db_offline`
   `dbmcli –d <database_name> -u <dbm_user>,<password> db_cold`
2. Conclude the backup history by performing a log backup or a complete data backup.
3. Execute the CLEAR LOG command in operational state COLD (ADMIN)
   `dbmcli –d <database_name> -u <dbm_user>,<password> -uUTL <dbm_user>,<password> util_execute clear log`
4. Restart the database instance.
5. Perform a complete data backup.

Subsequent builds of SAP DB version 7.4.03.10 are designed in such way that the log page number break occurs automatically and does not require user intervention.

7.8 Analysis of Severe Database Problems (System Error) Using X_DIAG(NOSE)

Severe database errorsare the system errors of the database (-9999 to –9000). These errors are logged in `knldiag` and `knldiag.err`.
For detailed information about how to analyze and correct individual system errors, see section 4.2. If the measures described there do not help further, you can use the X_DIAG tool (Microsoft Windows) or the X_DIAGNOSE tool (UNIX) to continue your analysis.

7.8.1 Analysis Files: `<page_type><log.number>.bad` or `<page_type><log.number>.cor`

The run directory and the **`DIAGHISTORY`** are again used as a starting point for an analysis here. If there are files in the specified directories with the suffix `*.cor` or `*.bad` that were created at the time of the error, you can use these for additional analysis with X_DIAGNOSE. These files contain a DUMP of the page that caused the error. The name of the file is made up of the type of the page, the logical page number,

and the suffix. You cannot read the files with an editor, but must use X_DIAGNOSE to analyze them instead.

Example of a dump of a data page: `d1015836.bad`

Procedure:

Call X_DIAGNOSE in the run directory or diaghistory directory without any additional parameters.
You can first create a **log file** (default: `diag.prt`) that is created in the current directory and will contain the analysis data in a readable format. You should choose a descriptive name for this log file, to facilitate the analysis, which is usually then performed by a developer. You should check whether there is already a file with this name in the current directory, as in this case, the new content is added to the file in the case of the name `diag.prt`, and the old file is overwritten in the case of other file names.

Next, choose ***3 Typebuf*** . Now specify the dump file as the **input file** (Example: `d1015836.bad`).
The dump can then be displayed using the option ***1 ALL***. In our example case, this is a NIL page:

```
TYPEBUF  7.2.5
d1015836.bad
--------------------------------------------------------------------------------
NIL 1048576  [page 0]

END OF FILE
```

In this case, this output alone provides little information. You or the SAP DB expert will usually provide information about whether a ***6 SCAN*** should be activated or should be further analyzed unformatted using ***7 NOSCAN***. You should also always consider this output together with `knldiag` and `knldiag.err`. This is to ensure that the circumstances that could have led to this problem are always taken into consideration.

7.8.2 Analyzing the Devspaces (Volumes) Directly

With X_DIAGNOSE, it is also possible to analyze the data, log, and system devspaces or volumes directly. However, this is usually done only by a SAP DB expert.
X_DIAGNOSE is also called without parameters in this case, and a **log file** specified. The devspace (volume) that is to be analyzed is specified as the ***INPUTFILE***.

Example:

```
ERR          54001  I/O    BAD DATA PAGE 1015836
ERR          54001  I/O    on DEVNO 2 DEV_OFFSET 22177
ERR          53016  I/O    /sapdb/SQ2/sapdata/DISKD0002
ERR          53021  B*TREE BAD FILE: 1015836 (ROOT)
ERR          51080  SYSERROR     -9026 BD Bad datapage
```

You can use the error message in `knldiag`, you can identify that the logical page 1015836 is on devspace (volume) number 2 (on **DEVNO 2** DEV_OFFSET 22177). You enter the name of the devspace (volume) in X_DIAGNOSE, not the number of the devspace (volume). You can determine which devspace (volume) corresponds to number 2 by choosing in the Database Manager GUI *Configuration*, if this is not output in `knldiag`.

Then activate a corresponding scan by choosing ***6 SCAN*** , which is usually specified by the developer, or continue the analysis by choosing ***7 NOSCAN*** .

You should not work with the option ***1 ALL*** on the devspaces (volumes), as the analysis file can become very large in this case, but rather restrict the analysis area using ***2 FROM/TO*** or ***3 GET PAGE***. Point ***4 EDIT PAGE*** is not discussed here, as in this context, it is not possible to edit a data page (change the data directly on the devspace or volume), although you might assume otherwise from this menu option.

When choosing the menu options ***FROM/TO*** and ***GET PAGE***, you should note that in this case you should specify the **physical block address** of the page and not the logical page number.
You can find the relevant information in `knldiag`.

Example:

```
ERR          54001  I/O    BAD DATA PAGE 1015836
ERR          54001  I/O    on DEVNO 2 DEV_OFFSET 22177
ERR          53016  I/O    /sapdb/SQ2/sapdata/DISKD0002
ERR          53021  B*TREE BAD FILE: 1015836 (ROOT)
ERR          51080  SYSERROR     -9026 BD Bad datapage
```

In our example, the logical page 1015836 is on devspace (volume) number 2, at the offset 22177. You should therefore specify ***PAGENO*** 22177 in X_DIAGNOSE.

This procedure is usually used to confirm theories about the cause of the error. Analysis the devspaces (volumes) directly makes it easier, for example, to justify hardware errors with hardware partners. In all of these cases, it is important that all diagnosis files are made available for further analysis.

7.9 Problem Analysis if the Database Can No Longer Be Transferred to the WARM (ONLINE) Operational State(Database operational state COLD (ADMIN))

Start Situation:
The database has crashed without any identifiable reason and cannot now be transferred to the operational state WARM (ONLINE) .

You can also extract important information from the database for finding the cause in the COLD (ADMIN) operational state.

7.9.1 Analyze Pages

If you know the affected pages (`knldiag`), they can be analyzed with X_DIAGNOSE directly on the devspaces (volumes) in the same way as described in section 7.8.2 .

7.9.2 Unloading the File Directory

The assignment of a database object to its root page number is stored in the file directory. You can use the file directory in the COLD (ADMIN) operational state to determine the name of the defective object, if you know the root page number. `knldiag` (`knldiag.err`) is again used to do this.

```
ERR          54001  I/O    BAD DATA PAGE 1015836
ERR          54001  I/O    on DEVNO 2 DEV_OFFSET 22177
```

```
ERR          53016  I/O    /sapdb/SQ2/sapdata/DISKD0002
ERR          53021  B*TREE BAD FILE: 1015836 (ROOT)
ERR          51080  SYSERROR     -9026 BD Bad datapage
```

In our example, a BAD DATA PAGE was reported for page 1015836. `knldiag` also provides us with the information that this page is a root page (B*TREE BAD FILE: 1015836 (ROOT)).

In the WARM (ONLINE) operational state, you can determine the name of the affected table using the ROOTS table (`SELECT * FROM roots WHERE root = 1015836`). It is not possible to access SQL tables in the COLD (ADMIN) operational state. You can only use X_DIAGNOSE to unload the file directory in the COLD (ADMIN) operational state.

To do this, call X_DIAGNOSE with the following parameters:
Call Under Microsoft Windows:

```
x_diag -d <database_name> -u <dbm_user>,<password>
```

The file directory is read in X_DIAGNOSE by choosing ***4 TYPEDATA*** . To view the entries in the file directory, the relevant format must be activated by choosing ***6 SCAN*** . You normally select ***K(key)*** here - in special cases, also ***R(Record)***. By choosing ***2 GET FILE DIRECTORY***, you can select the desired file directory. By choosing ***1 FILE DIRECTORY***, all entries are read for the base objects. The blobs (LONG values > 8 KB) are stored in the LONG file directory. You can read these by choosing ***2 LONG DIRECTORY***. However, this is more unusual. You can log the file directory to the file `util.prt` by choosing the function key <F5 NOHOLD>. `util.prt` is also created in the current directory.

The output of the file directory to the `util.prt` file can become very large (depending on the size of the database). It is useful to load the file in an editor and search for the root page number.

Example:

```
 40: (pos 01665) root 1015836  TABID 0-02C3
 41: (pos 01701) root 28854  TABID 0-02C5
 42: (pos 01737) root 67811  TABID 0-02C6
 43: (pos 01773) root 375    TABID 0-02F1
 44: (pos 00441) root 34     SCOL 0-A3
 45: (pos 00549) root 30     SCOL 0-A9
 46: (pos 00945) root 171    SCOL 0-01EE
```

In our example, we find the root 1015836. It belongs to a database object with the TABID 0-02C3. (Note: SCOL means that this is a Short Column File - LONG < 8 KB.)

Determining the Object name Using the TABID

You can now determine the table name using X_DIAGNOSE and the TABID .
Call X_DIAGNOSE again here, specifying the database instance and the DBM operator.

```
x_diag -d <database_name> -u <dbm_user>,<password>
```

You can branch to a special diagnosis menu by choosing ***1 DIAGNOSE***. You can determine the table name by choosing ***12 GET TABLENAME*** and entering the TABID (02C3).

In some cases, it is possible that you will receive a ROW NOT FOUND or UNKNOWN TABLE error after specifying the TABID. In this case, you should call X_DIAGNOSE as a different user, such as the SYSDBA user, and enter the TABID here.

You can determine what needs to be done now using the information about which database object is the error source. For example, you would follow a different procedure for an index or a table, the contents of which can be generated from other tables, than if the source is a table, the content of which cannot be generated. In the latter case, only a restore of the dataset could solve the problem.

7.9.3 Restart Record

You can also use X_DIAGNOSE to extract the restart record in the COLD (ADMIN) operational state.

```
x_diag -d <database_name> -u <dbm_user>,<password>
```

In versions below 7.4, the restart record is on the system devspace and also on the log devspace (volume). By choosing ***1 DIAGNOSE*** , ***6 LOGSCAN*** , ***1 LOG*** , and specify the log devspace number, you can access the restart record directly with ***15 EDIT RESTART RECORD***. It is not possible to change the restart record in this case either, even if the menu option suggests that it is.

As of version 7.4, the restart record is stored on the first data volume and can be extracted with X_DIAGNOSE in the same way as a data page (section 7.9.1).

7.10 Problems with Backup/Restore

7.10.1 General Problems

A backup or a restore can fail for a wide variety of reasons. The customer often reports only an I/O error (-903 or -902). It is always important to determine the exact error message. The following files are very helpful in doing so: `dbm.prt`, `dbm.knl`, `dbm.utl`, `knldiag`.
If an error occurs at the start of the thread or at a fork, it is usually the case that operating system parameters are set to insufficient values (such as the maximum number of processes/threads per user). In the case of the error “Access denied“, you should check the authorizations on the backup medium and the devspaces (volumes).

Example of a Problem at the Fork on Solaris:

`dbm.knl`*:*

Only error -903 is displayed in the backup history.

```
3B1D81B80002|DAT_00066|RESTORE  |2001-05-28 12:32:01|2001-05-28 12:32:01|2001-06-06
09:05:04|2001-06-06 09:05:19|  18851962|          |NO |DLT0
|         8|         0|      -903|                                        |
```

`knldiag`*:*

In `knldiag`, on the other hand, you can see the exact fields. There was a problem at asynopen of devspace (volume) DISKD06:

```
06-06 09:04:56    10     11597 IO       Open '/dev/rmt/0c' successful, fd: 21
06-06 09:05:03    47     12821 TASKING  Thread 47 starting
06-06 09:05:04    47     11565 startup  DEVi started
```

```
06-06 09:05:04    14     11000 vasynopen '/dev/rmt/0c' devno 32 T5 succeeded
06-06 09:05:04    14     52101 RESTORE  Filetype: tape (rewind)
06-06 09:05:04    14     11000 vdevsize '/adabas/TNT/sapdata/DISKS01', 2768 succeeded
06-06 09:05:04    10     11597 IO       Open '/adabas/TNT/sapdata/DISKS01' successful,
fd: 22
06-06 09:05:04    48     12821 TASKING  Thread 48 starting
06-06 09:05:04    48     11565 startup  DEVi started
06-06 09:05:04    10     11597 IO       Open '/adabas/TNT/sapdata/DISKS01' successful,
fd: 24
06-06 09:05:04    49     12821 TASKING  Thread 49 starting
06-06 09:05:04    49     11565 startup  DEVi started
06-06 09:05:04    14     11000 vattach  '/adabas/TNT/sapdata/DISKS01' devno 1 T5
succeeded
06-06 09:05:04    14     11000 vdevsize '/adabas/TNT/sapdata/DISKD01', 255999 succeeded
06-06 09:05:04    10     11597 IO       Open '/adabas/TNT/sapdata/DISKD01' successful,
fd: 25
06-06 09:05:04    50     12821 TASKING  Thread 50 starting
06-06 09:05:04    50     11565 startup  DEVi started
06-06 09:05:04    10     11597 IO       Open '/adabas/TNT/sapdata/DISKD01' successful,
fd: 26
06-06 09:05:04    51     12821 TASKING  Thread 51 starting
06-06 09:05:04    51     11565 startup  DEVi started
06-06 09:05:04    14     11000 vattach  '/adabas/TNT/sapdata/DISKD01' devno 2 T5
succeeded
…
06-06 09:05:19    14     11000 vasynopen '/adabas/TNT/sapdata/DISKD05' devno 20 T5
succeeded
06-06 09:05:19    10     12822 TASKING  Thread 82 joining
06-06 09:05:19    82     12821 TASKING  Thread 82 starting
06-06 09:05:19    10     11597 IO       Open '/adabas/TNT/sapdata/DISKD06' successful,
fd: 63
06-06 09:05:19    10 ERR 11000 d0_aopen Error during fork, Too many open files
06-06 09:05:19    14     11000 vasynopen '/adabas/TNT/sapdata/DISKD06' T5 failed
06-06 09:05:19    14     11000 vasynclos '/dev/rmt/0c' devno 32 T5
06-06 09:05:19    10     12822 TASKING  Thread 47 joining
06-06 09:05:23    47     11566 stop     DEVi stopped
06-06 09:05:23    14     11000 vasynclos '/adabas/TNT/sapdata/DISKD01' devno 16 T5
06-06 09:05:23    10     12822 TASKING  Thread 73 joining
06-06 09:05:23    73     11566 stop     DEVi stopped
…
06-06 09:05:23    69     11566 stop     DEVi stopped
06-06 09:05:23    14     11000 vdetach  '/adabas/TNT/saplog/DISKLB1' devno 12 T5
06-06 09:05:23    10     12822 TASKING  Thread 70 joining
06-06 09:05:23    70     11566 stop     DEVi stopped
06-06 09:05:23    10     12822 TASKING  Thread 71 joining
06-06 09:05:23    71     11566 stop     DEVi stopped
06-06 09:05:23    14 ERR 52012 RESTORE  error occurred, basis_err 3700
```

Solution:

The error message at the fork (Too many open files) indicates that the UNIX parameter 'maximum of file descriptors' is set to too small a value. You can determine this parameter with the command 'ulimit -n'.

7.10.2 Problems with External Backup Tools

For information about how a backup with external backup tools works, see the documentation *External Backup Tools: SAP DB* on www.sapdb.org.

The file `dbm.ebp` is required if there are problems with external backup tools. This file contains the most important information, if external backup tools are used. In this way, for example you can see the configuration parameters for the backup tool and exactly where the error occurs (at the backup tool or in the database kernel). However, this file is overwritten every time a new DBM Server is started and this communicates with the external backup tool (for example, in the case of the Database Manager CLI command `backup_ext_ids_get`). A new DBM Server is always started when you execute a

new Database Manager CLI command, WITHOUT being in a Database Manager CLI session.

Problems with external backup tools are often due to incorrect configuration. The customer should therefore also provide the various configuration files.

If an error occurs only during the “post-backup steps”, the Database Manager CLI reports an error and the error is also logged in the files `dbm.ebp` and `dbm.prt` . However, you will not find any errors in the `knldiag` and `dbm.knl` files. Even the Database Manager GUI shows that the backup was “successful”.
After the backup is created, the DBM Server attempts to determine the external backup ID (EBID) that belongs to the backup that has just been created. From the database’s point of view, the backup is already successfully completed at this point, which is why no errors are logged in the files `dbm.knl` and `knldiag`. However, if it is not possible to determine the corresponding EBID, it is not certain that the backup that has just been created would be available for a restore. The Database Manager CLI therefore reports the error and logs it in the files `dbm.prt` and `dbm.ebp`. It must then be checked carefully to see why the EBID could not be determined.

Example of Problems when Determining the EBID with NetWorker:

`dbm.prt`*:*

You can already identify in this file, from the message “ERR_POSTOP”, that the error occurred during the post-backup operations:

```
08-09 14:21:08 0x000013dd              0 DBM      command backup_start NSR_D LOG
08-09 14:22:37 0x000013dd ERR     -24921 DBM      ERR_POSTOP: error while finishing
backup operation
               0x000013dd ERR     -24921 DBM      Could not get external backup ID's from
the backup tool.
```

`dbm.knl`*:*

As the backup was completed successfully from the point of view of the database, there are no error messages in the backup history (Return code 0):

```
3B7280360006|LOG_00336|SAVE WARM|2001-08-08 03:21:08|2001-08-09 14:21:10|2001-08-09
14:21:10|2001-08-09 14:22:28|         0|     22174|   |NSR_D
|     22192|         1|         0|                                        |
```

`dbm.ebp`*:*

There is information about the parameters set, the checks, and the backup preparations at the start of the file. All of these steps were successful:

```
Using connection to Legato's NetWorker with save, recover and mminfo.

Checking existence and configuration of NetWorker.
    Using configuration variable 'NSR_ENV' = '/nsr/sapdb/env' as path of NetWorker's
configuration file.
    Setting environment variable 'NSR_ENV' for the path of NetWorker's configuration file
to configuration value '/nsr/sapdb/env'.
    Found NetWorker setting for 'NSR_HOME': '/usr/opt/networker/bin'.
    Found NetWorker setting for 'NSR_HOST': 'dehsor1e.hbg.de.origin-srv.com'.
    Found NetWorker setting for 'NSR_POOL': 'ARCA014'.
    Found NetWorker setting for 'NSR_EXPI': 'Month'.
    Using NetWorker programs:
        '/usr/opt/networker/bin/save'
        '/usr/opt/networker/bin/recover'
        '/usr/opt/networker/bin/mminfo'
```

```
Check passed successful.

Checking medium.
Check passed successfully.

Preparing backup.
    Constructed NetWorker's directives file '/tmp/.nsr' successfully.
    Constructed NetWorker call '/usr/opt/networker/bin/save -v -s dehsor1e.hbg.de.origin-
srv.com -b ARCA014 -e Month -N AT1 -l 1 -f /tmp/.nsr /tmp/lega_dat'.
    Created temporary file '/tmp/aaaaaeEDa' as output for NetWorker.
    Created temporary file '/tmp/baaaaeEDa' as error output for NetWorker.
    Waiting 1 second ... Done.
Prepare passed successfully.

Creating pipes for data transfer.
    Creating pipe '/tmp/lega_dat' ... Done.
All data transfer pipes have been created.

Starting database action for the backup.
    Requesting 'SAVE LOG QUICK TO '/tmp/lega_dat' PIPE BLOCKSIZE 8 MEDIANAME 'NSR_D''
from db-kernel.
The database is working on the request.

Waiting until database has prepared the backup.
    No reply from database available.
    Waiting 1 second ... Done.
    Asking for state of database.
    Got the following reply from db-kernel:
        SQL-Code              :0
        Date                  :20010809
        Time                  :00142110
        Database              :AT1
        Server                :DEHSAT11
        KernelVersion         :Kernel    7.2.5    Build 000-000-232-798
        PagesTransfered       :0
        PagesLeft             :0
        DevicesUsed           :1
        DatabaseID            :DEHSAT11:AT1_20010808_032108
        Redo Transactions Read:0
        Redo Transactions Done:0
    Checking for reply of backup request.
    No reply from database available.
The database has prepared the backup successfully.
```

Then the backup is actually started, and neither the backup tool nor the database report any errors. The messages “No reply from database available” are normal and DO NOT mean that there is a problem. While the backup is running, DBM Server regularly queries the status of the backup tool and of the database. The utility task is occupied during the backup, and this is used for this test. While the connect to the utility task does not work, the backup is not yet finished, and this message is displayed.

```
Starting NetWorker.
    Starting NetWorker process '/usr/opt/networker/bin/save -v -s dehsor1e.hbg.de.origin-
srv.com -b ARCA014 -e Month -N AT1 -l 1 -f /tmp/.nsr /tmp/lega_dat >>/tmp/aaaaaeEDa
2>>/tmp/baaaaeEDa'.
    Process was started successfully.
NetWorker has been started successful.

Waiting for end of the backup operation.
    Checking backup tool.
    The backup tool is running.
    No reply from database available.
    Waiting 5 seconds ... Done.
…
    Checking backup tool.
    The backup tool is running.
    No reply from database available.
```

```
    Waiting 13 seconds ... Done.
    Checking backup tool.
        The archiving tool process has finished work with return code 0.
    The backup tool is not running.
    Reply from database available.
    Receiving Reply.
    Got the following reply from db-kernel:
        SQL-Code              :0
        Date                  :20010809
        Time                  :00142110
        Database              :AT1
        Server                :DEHSAT11
        KernelVersion         :Kernel    7.2.5    Build 000-000-232-798
        PagesTransfered       :22192
        PagesLeft             :0
        Volumes               :1
        MediaName             :NSR_D
        Location              :/tmp/lega_dat
        Label                 :LOG_00336
        FirstLogPageNo        :0
        LastLogPageNo         :22174
        DBStamp1Date          :20010808
        DBStamp1Time          :00032108
        DBStamp2Date          :20010809
        Time          :00142110
        BDPageCount           :22174
        DevicesUsed           :1
        DatabaseID            :DEHSAT11:AT1_20010808_032108
    Checking backup tool.
    The backup tool is not running.
The backup operation has ended.
```

After the backup has been successfully performed (without error messages), the log files are updated. To do this, the DBM Server must determine the label and the EBID of the backup. In this case, it can determine the label, however Networker does not return an EBID for this label:

```
Updating external backup file.
    Using '/abis/sapdb/data/wrk/AT1/dbm.knl' as backup history.
        Looking for LOG_00336 with DB Stamp1 2001-08-08 03:21:08.
        Found 1 SaveID. It was 3B7280360006|LOG_00336.
    Got '3B7280360006|LOG_00336' as save identifier.
    Using '/abis/sapdb/data/wrk/AT1/dbm.ebf' as external backup file.
    Requesting SaveID from NetWorker.
     Could not get the External Save ID for the last save.
    Have encountered error -24921:
        Could not get external backup ID's from the backup tool.
```

This is then returned to the client as a response:

```
    Constructed the following reply:
        ERR
        -24921,ERR_POSTOP: error while finishing backup operation
        Could not get external backup ID's from the backup tool.

Could not update external backup file.
```

At the end of the `dbm.ebp` file, the error messages of the backup tool are logged. However, the Networker output does not contain ANY error messages in this example. The command `mminfo` that is used to determine the EBID therefore returned no errors, but did not return the desired backup:

```
Cleaning up.
    Removing data transfer pipe /tmp/lega_dat ... Done.
    Moving output of NetWorker to this file.
    ---------- Begin of output of NetWorker (/tmp/aaaaaeEDa)----------
ase_init retcode = 7
    ---------- End of output of NetWorker (/tmp/aaaaaeEDa)----------
```

```
    Removed NetWorker's temporary output file '/tmp/aaaaaeEDa'.
    Moving error output of NetWorker to this file.
    ---------- Begin of error output of NetWorker (/tmp/baaaaeEDa)----------
save: got prototype for /
save: got prototype for /
…
save: got prototype for /abis/ediadm/psp/
chdir(/tmp/)
Name=`/tmp/lega_dat', name=`/tmp/lega_dat', fname=`lega_dat'
save: found protofile spec for /:
  mntasm : data
save: found protofile spec for /tmp/:
  mntasm : install
save: /tmp/.nsr parsed
walk(/tmp/lega_dat, lega_dat)
matched internal `rawasm' on `lega_dat' for `/tmp/lega_dat'
rawasm -s /tmp/lega_dat
Connecting directories...
dir: /tmp/, fstype: 3, fsid: 0
save: found protofile spec for /:
  mntasm : data
save: found protofile spec for /tmp/:
  mntasm : install
save: /tmp/.nsr parsed
matched internal `rawasm' on `lega_dat' for `/tmp/'
uasm -s /tmp/
/tmp/ file opened for AdvFS extended attrs
dir: /, fstype: 3, fsid: 0
save: found protofile spec for /:
  mntasm : data
matched internal `mntasm' on `usr' for `/'
matched internal `mntasm' on `proc' for `/'
matched internal `mntasm' on `abis' for `/'
matched internal `mntasm' on `data' for `/'
uasm -s /
/ file opened for AdvFS extended attrs

save: AT1 level=1, 177 MB 00:01:18      3 files
    ---------- End of error output of NetWorker (/tmp/baaaaeEDa)----------
    Removed NetWorker's temporary error output file '/tmp/baaaaeEDa'.
    Removed NetWorker's directives file '/tmp/.nsr'.
Have finished clean up successfully.
```

Solution:

To find out why the backup that has just been created is not returned when the command `mminfo` is run, you should first run the command `backup_ext_ids_get` and `backup_ext_ids_list` in Database Manager CLI. You must check whether the desired backup is displayed there. In most cases, the backup is also not displayed there. You can also run the `mminfo` command, which is now logged in the file `dbm.ebp`, manually and vary the parameters. However, in most cases, it is a problem of the `mminfo` command, which is delivered by Legato together with NetWorker. The customer should therefore contact Legato Support.

Example of Incorrect Configuration with BACKINT:

`dbm.prt`*:*

There are no helpful error messages in this file, it is simply noted that the backup tool reported an error:

2002-03-25 17:08:01 0x00000043 0 DBM command backup_save "back_first" DATA RECOVERY

```
2002-03-25 17:08:04 0x00000043 ERR     -24920 DBM      ERR_BACKUPOP: backup operation was
unsuccessful
                    0x00000043 ERR     -24920 DBM      The backup tool failed with 2 as
sum of exit codes.
```

dbm.ebp:

In this file, you can see right at the beginning that there is a problem with the configuration file. Several parameters are listed for which the keywords are not correct:

```
2002-03-25 17:08:01
Using environment variable '%TEMP%' = 'C:\TEMP' as directory for temporary files and
pipes.
Using connection to Backint for SAP DB Interface.

2002-03-25 17:08:01
Checking existence and configuration of Backint for SAP DB.
```

Using configuration variable 'BSI_ENV' = 'd:\sapdb\data\config\backint.conf' as path of the configuration file of Backint for SAP DB.

```
    Setting environment variable 'BSI_ENV' for the path of the configuration file of
Backint for SAP DB to configuration value 'd:\sapdb\data\config\backint.conf'.
    Reading the Backint for SAP DB configuration file
'd:\sapdb\data\config\backint.conf'.
```

Found keyword 'BACKINT' with value 'D:\sapdb\SDB\DB\bin\backint.exe'.

```
       The following line of the Backint for SAP DB configuration file does not start
with a proper keyword and is ignored:

        Found keyword 'INPUT' with value 'C:\TEMP\backint4SAPDB.in'.
        Found keyword 'OUTPUT' with value 'C:\TEMP\backint4SAPDB.out'.
        Found keyword 'ERROROUTPUT' with value 'C:\TEMP\backint4SAPDB.err'.
        The following line of the Backint for SAP DB configuration file does not start
with a proper keyword and is ignored:
```

The following line of the Backint for SAP DB configuration file does not start with a proper keyword and is ignored:

```
            ;PARAMETERFILE C:\SAPDB\WRK\TST\backint.par
        The following line of the Backint for SAP DB configuration file does not start
with a proper keyword and is ignored:

        Found keyword 'TIMEOUT_SUCCESS' with value '600'.
        Found keyword 'TIMEOUT_FAILURE' with value '300'.
        The following line of the Backint for SAP DB configuration file does not start
with a proper keyword and is ignored:

        The following line of the Backint for SAP DB configuration file does not start
with a proper keyword and is ignored:
            ;ORIGINAL_RUNDIRECTORY C:\SAPDB\wrk\P1
```

Finished reading of the Backint for SAP DB configuration file.

The messages that show which programs, files, and parameters are used for Backint for SAP DB are then displayed:

```
    Using 'D:\sapdb\SDB\DB\bin\backint.exe' as Backint for SAP DB program.
    Using 'C:\TEMP\backint4SAPDB.in' as input file for Backint for .
    Using 'C:\TEMP\backint4SAPDB.out' as output file for Backint.
    Using 'C:\TEMP\backint4SAPDB.err' as error output file for Backint.
    Using no parameter file for Backint for SAP DB.
    Using '600' seconds as timeout for Backint in the case of success.
    Using '300' seconds as timeout for Backint in the case of success.
    Using 'D:\sapdb\data\wrk\SDB\dbm.knl' as backup history of a database to migrate.
    Using 'D:\sapdb\data\wrk\SDB\dbm.ebf' as external backup history of a database to
migrate.
Check passed successful.
```

Then there are additional checks and preparations:

```
2002-03-25 17:08:02
Checking medium.
```

```
Check passed successfully.

2002-03-25 17:08:02
Preparing backup.
    Setting environment variable 'BI_CALLER' to value 'DBMSRV'.
    Setting environment variable 'BI_REQUEST' to value 'NEW'.
    Setting environment variable 'BI_BACKUP' to value 'FULL'.
    Constructed Backint for SAP DB call 'D:\sapdb\SDB\DB\bin\backint.exe -u SDB -f backup
-t file -i C:\TEMP\backint4SAPDB.in -c'.
    Created temporary file 'C:\TEMP\backint4SAPDB.out' as output for Backint for SAP DB.
    Created temporary file 'C:\TEMP\backint4SAPDB.err' as error output for Backint for
SAP DB.
    Writing '\\10.3.10.248\f$\backup\savedata_original #PIPE' to the input file.
Prepare passed successfully.

2002-03-25 17:08:02
Starting database action for the backup.
    Requesting 'SAVE DATA QUICK TO '\\10.3.10.248\f$\backup\savedata_original' FILE
BLOCKSIZE 8 NO CHECKPOINT MEDIANAME 'back_first'' from db-kernel.
The database is working on the request.

2002-03-25 17:08:02
Waiting until database has prepared the backup.
    No reply from database available.
    Waiting 1 second ... Done.
    Asking for state of database.
    Got the following reply from db-kernel:
        SQL-Code              :0
        Date                  :20020325
        Time                  :00170802
        Database              :SDB
        Server                :BASAPCONT
        KernelVersion         :Kernel    7.3.0    Build 017-000-078-540
        PagesTransfered       :64
        PagesLeft             :207
        BDPageCount           :271
        DevicesUsed           :1
        DatabaseID            :BASAPCONT:SDB_20020321_183637
        Max Used Data Page    :348
        Redo Transactions Read:0
        Redo Transactions Done:0
    Checking for reply of backup request.
    No reply from database available.
The database has prepared the backup successfully.
```

Finally, the backup is started:

```
2002-03-25 17:08:03
Starting Backint for SAP DB.
    Starting Backint for SAP DB process 'D:\sapdb\SDB\DB\bin\backint.exe -u SDB -f backup
-t file -i C:\TEMP\backint4SAPDB.in -c >>C:\TEMP\backint4SAPDB.out
2>>C:\TEMP\backint4SAPDB.err'.
    Process was started successfully.
Backint for SAP DB has been started successfully.
```

An error is logged after only a short time. The backup tool has ended with a non-zero return code. The following return code of the database is then unimportant, as the first error in this file is decisive:

```
2002-03-25 17:08:03
Waiting for end of the backup operation.
    Checking backup tool.
        The backup tool process has finished work with return code 2.
```

The backup tool is not running.

```
    Reply from database available.
    Receiving Reply.
    Got the following reply from db-kernel:
        SQL-Code              :0
        Date                  :20020325
        Time                  :00170802
```

```
        Database              :SDB
        Server                :BASAPCONT
        KernelVersion         :Kernel    7.3.0    Build 017-000-078-540
        PagesTransfered       :304
        PagesLeft             :0
        Volumes               :1
        MediaName             :back_first
        Location              :\\10.3.10.248\f$\backup\savedata_original
        Label                 :DAT_00016
        IsConsistent          :true
        FirstLogPageNo        :544
        DBStamp1Date          :20020325
        DBStamp1Time          :00170802
        BDPageCount           :271
        DevicesUsed           :1
        DatabaseID            :BASAPCONT:SDB_20020321_183637
        Max Used Data Page    :348
    Checking backup tool.
    The backup tool is not running.
The backup operation has ended.
```

This is then reported as the result to the client:

```
2002-03-25 17:08:03
Filling reply buffer.
    Have encountered error -24920:
        The backup tool failed with 2 as sum of exit codes.
    Constructed the following reply:
        ERR
        -24920,ERR_BACKUPOP: backup operation was unsuccessful
        The backup tool failed with 2 as sum of exit codes.
Reply buffer filled.
```

The error messages of the backup tool are then logged at the end of the `dbm.ebp` file. This determines that the parameters “staging area”, “path of backint’, “input file”, “output file”, and “Error output file’ were not correctly configured for BACKINT FOR ORACLE. These parameters were already correctly listed for Backint for SAP DB at the start of the file.

```
2002-03-25 17:08:03
Cleaning up.
    Moving output of Backint for SAP DB to this file.
    ---------- Begin of output of Backint for SAP DB (C:\TEMP\backint4SAPDB.out)---------
-
    ---------- End of output of Backint for SAP DB (C:\TEMP\backint4SAPDB.out)----------
    Removed Backint for SAP DB's temporary output file 'C:\TEMP\backint4SAPDB.out'.
    Moving error output of Backint for SAP DB to this file.
    ---------- Begin of error output of Backint for SAP DB (C:\TEMP\backint4SAPDB.err)---
----
No staging area is defined in the parameter file.
The path of backint is not defined in the parameter file.
The name of the input file for backint is not defined in the parameter file.
The name of the output file for backint is not defined in the parameter file.
The name of the error output file for backint is not defined in the parameter file.
    ---------- End of error output of Backint for SAP DB (C:\TEMP\backint4SAPDB.err)-----
-----
    Removed Backint for SAP DB's temporary error output file 'C:\TEMP\backint4SAPDB.err'.
    Removed the Backint for SAP DB input file 'C:\TEMP\backint4SAPDB.in'.
Have finished clean up successfully.
```

Solution:
Specify the incorrect parameters in the relevant parameter files. Note that two backup tools are used here: Backint for SAP DB and Backint for Oracle. BOTH require a configuration file, as described in the documentation.

7.11 Problems with the Tools SAPDBINSTALL, SDBINST, and SDBUPD

These tools are used to install the database software or to install a new version. The SAPDBINSTALL tool is used in Version 7.2.04 Build x (x<15). The SDBINST tool is used in versions 7.2.04 Build x (x >= 15), 7.2.05, 7.3, and 7.4.01. In Version 7.4.02, the SDBUPD tool is used to import a new version, however, SDBINST is still used for a new installation and for the update 7.4.01 → 7.4.02.
The relevant tool provides a large amount of useful information at the command prompt (or in the UNIX shell). If this information is not sufficient, you can search for the cause of the error in the log `SAPDB*<date><time>.LOG` of the tool. It is stored in the `<independent_data_path>/wrk`.
When importing a SAP DB patch using the SAPDBINSTALL or SDBINST tool (as of version 7.2.04 Build 15), there are often problems under Microsoft Windows when overwriting DLLs. The X Server is often not stopped. You can check this, for example, using the Task Manager.

7.12 Problems with Overwriting DLLs (Microsoft Windows)

If DLLs or Executables cannot be overwritten during an update of the database software under Microsoft Windows, this is due to the fact that the relevant file is still being used -that is, that a program has started the relevant executable or loaded the relevant DLL. You can use the HANDLEEX to find out which program is responsible for the problems.

7.13 Uninstallation

7.13.1 Uninstalling the Database Software

There is no tool with which the database software can be uninstalled, until Version 7.4.02 Build 06. This can lead to problems, if there is already SAP DB software on a host, but the system is to be restructured, and the database software is to be installed in a different directory. The SDBINST tool does not allow you to specify an `<independent_data_path>` path different from the one that already exists.
In these versions, the uninstallation can be performed manually.
If the old software was not uninstalled correctly, registrations that prevent a new installation will still exist. You can remove these registrations. If occur any problems, contact a SAP DB expert.
If, in addition to the software < 7.4.02 Build 07, there is also already a newer software version on the host, the old software can be uninstalled using the SDBUNINST tool delivered with the newer software. However, in some cases, only the registrations are removed in this case, and the files must then be deleted manually.

As of Version 7.4.02 Build 07, the uninstallation of the software can be performed using SDBUNINST.
The database installation consists of various SAP DB software components that you can display with the command `sdbuninst -l`.

Example:

```
PCR 7301            /sapdb/programs      7.3.01.01              valid
```

```
PCR 7401            /sapdb/programs      7.4.01.19              valid
PCR 7240            /sapdb/programs      7.2.04.17              valid
PCR 7402            /sapdb/programs      7.4.02.07              valid
Server Utilities    /sapdb/programs      7.4.02.07    64 bit    valid
PCR 7250            /sapdb/programs      7.2.05.18              valid
Database Kernel     /sapdb/ZW6/db        7.4.02.07    64 bit    valid
Database Kernel     /sapdb/LW6/db        7.2.05.18              valid
Base                /sapdb/programs      7.4.02.07    64 bit    valid
PCR 7300            /sapdb/programs      7.3.00.21              valid
APO COM             /sapdb/ZW6/db/sap    3.0A.34      64 bit    valid
```

You find a description of the software components in the documentation *Installation Manual: SAP DB*.

The complete installation of all components is noted with `Base`. If you specify the option `sdbuninst -all`, **all** components, starting from Base, are deleted.
We therefore recommend that you delete the components individually by specifying the name of the corresponding package.
To delete the installation or to delete individual components, you must run SDBUNINST as the root user.

Example:

```
sdbuninst –package "Database Kernel"
```

As there could be multiple installations on a single host, the system prompts you as to which installation of the specified package (such as Database Kernel) should be deleted. All installations of Database Kernel are listed and you then give the SDBUNISNT tool the number of the installation that is to be deleted.

Example:

```
0: Database Kernel  in  /sapdb/ZW6/db        7.4.02.07    64 bit
1: Database Kernel  in  /sapdb/LW6/db        7.2.05.18
2: none
```

7.13.2 Uninstallation of the Database Instance

- **Version 7.2**:
  Database Manager GUI, menu option *Instance → Drop*
  Database Manager CLI, command
  `dbmcli –d <database_name> -u <dbm_user>,<password>`
  `db_drop [WITHOUTFILES]`
  The database instance must also be in the OFFLINE operational state in this case.

8 Problems with OLTP Databases

8.1 Performance Problems

- **Performance Problems when Executing Transactions**:
  Use the optimizer statistics. Performance problems can also be caused by incorrect statistics. For more information see documentation *Optimizer: SAP DB 7.4*.

With this information, you can decide for example whether an additional index would speed up processing.

- **General Performance Problems**:
  - You should first check the cache hitrates using Database Manager:
    The DATA CACHE HITRATE should be at least 99%,
    the CONVERTER CACHE HITRATE 100%.
  - Database Analyzer:
    This tool performs the function of the X_WIZARD tool as of Version 7.4.02. For more information see documentation *Database Analyzer: SAP DB*.

9 Information About Version 7.4

9.1 Terms

A few new terms were introduced as of Version 7.4:

Old	New
Devspace	Volume
COLD	ADMIN
WARM	ONLINE

9.2 System Devspace and Converter

As of Version 7.4, there is no longer a system devspace. In the versions before 7.4, the system devspace contains the restart record and the converter.
As of version 7.4, the restart record is on the first data volume.
The converter is organized as a B* tree distributed on all data volumes.

9.3 New Converter

In the versions before 7.4, the converter was organized as an array of 8 KB pages. This had the disadvantage that during a system copy, the new database had to be configured to exactly the correct size to allow the last logically assigned page number in the converter to be stored.
This logical page number (**LAST DATA PNO)** can be very high due to temporary pages, even if the net usage of the database (with permanently filled pages) is not as high. In this case, a backup has far fewer pages than the last assigned logical page number. In the case of a system copy, however, the new system must be able to provide space for the last assigned page number.

As of version 7.4, the converter is organized as a B* tree, meaning that it is now possible to reduce the database to the net page usage. The **LAST DATA PNO** is now irrelevant.

9.4 Logging / Savepoint / Checkpoint in 7.4

As of Version 7.4, there are no longer any Checkpoints. The system continues to perform Savepoints at regular intervals. When doing this, it writes all changed pages from the cache to the data area.

As of Version 7.4, the Undo Log Files (Before Images), among other things, are now also stored in the data cache. As these are also permanently stored in the data area by a savepoint, the database instance can be restored at a restart in a **transaction-consistent** state **(at the time of the last savepoint)** (This is possible even without the associated log area, as the log area is not required to rollback the open transactions).

Undo Log File: In undo log administration, every change transaction creates an undo log file for itself, in which the undo log entries (before images) are written. Every undo log entry contains an undo log sequence number, which begins with 0. As of version 7.4, the before images are stored in the data cache. Storing the before images (undo log files) in the data area means that it is possible to create a transaction-consistent database that reflects the status of the database system at the time of the last savepoint, using only the information in the data area. The deletion of the undo log files is performed in the OLTP environment at the end of a transaction (by the transaction itself).

Transaction File: The data of all open transactions is written in the transaction file at the time of a savepoint. The transaction file is an internal database file. The transaction file is stored in the data area.

Transaction List: At a restart, the transaction list is recreated using this transaction file of the Log Reader. The transaction list is an internal database list of all open change transactions. The transaction list is stored in the main memory.

All change transactions are available globally in the database using this transaction list. There is a transaction entry in the transaction list for each transaction. Among other things, the transaction entry contains references to redo and undo log entries, if they exist.

For detailed information about Logging 7.4, see documentation *The SAP DB Database System →Log Concept*.

9.5 New Mirroring of the Log Area in 7.4

If customers are using hardware-based RAID-5 or RAID-1 systems, they can enter the log mode SINGLE (single log area), as these systems offer sufficient operation security for production systems. If the option of hardware-based mirroring is not available, the log mode DUAL (mirrored log area) can be used.

In the case of mirroring using database functions, the database instance uses two log areas in parallel.

- The system writes to both log areas in parallel.

- **However, at a restart, or during the log backup, the system reads the log entries only from one log area, the primary log area.**

Both log areas can only be overwritten once the log entries of the primary log area have been backed up; that is, a log backup has been performed.
If there are access problems to the primary or secondary log area, the affected log volume is marked as BAD and**, unlike the versions before 7.4, the database instance is transferred to the OFFLINE operational state.**
The defective log volume of a log area can be restored in the **ADMIN operational state**. When you do this, the **complete** contents of the relevant log volume are copied from the other log area to the defective log volume.
This ensures that it is not necessary to restore the database instance after a disk error that destroys the contents of a log volume, as the contents of the log volume were mirrored.
For more information see documentations *The SAP DB Database System→Security Concepts →Log Settings* and *Log Settings – New Developments in the SAP DB Versions 7.3 to 7.4*.

10 XUSER Data Maintenance

If connect problems occur, you should basically check the XUSER data.
If the XUSER tool is not available, or if this tool cannot be used due to function keys that do not work, you can also maintain the XUSER data using the following Database Manager CLI commands.

Listing All Available XUSER Entries:

```
dbmcli –ux <default_user>,<password> -ul
```

Example*:* `dbmcli –ux sapr3,sap –ul`

Changing a Specific USER Key:

```
dbmcli -us <user>,<password> -d <database_name>
-n <database_server> -uk <user_key>
```

Example: `dbmcli -us dbm,dbm -d LCA -n p43273 -uk c`

The user name and password are specified with the option `–us`, which should be stored for the user key that was specified with `–uk`. The options `–n` and `–d` are evaluated at the same time, and the values are stored in the XUSER data (for database server and database name), and are also read and used if -U is used for authorization using XUSER data.
If the user key is already occupied, you must specify the user stored there with the option `–u`, to authorize yourself. Alternatively, you can use -ux to authorize yourself with the DEFAULT user.

Changing Specific XUSER Parameters:

```
dbmcli -uk <user_key> -us <user>,<password>
-d <database_name> -n <database_server> -up
<specification>
<specification> ::= <param>=<value>;[<specification>] | <param>=<value>;
<param>         ::= SQLMODE | TIMEOUT | CACHELIMIT | ISOLATION |
DBLOCALE
```

```
<value>         ::= valid value for <param>
                    in the case of SQLMODE, only
                    INTERNAL,ANSI,DB2,ORACLE,SAPR3
                    in the case of ISOLATION, only 0,1,2,3,10,15,20,30
```

Example:

```
dbmcli -us dbm,dbm -d LCA -n p43273 -uk c
-up SQLMODE=INTERNAL;TIMEOUT=42;
```

Ensure that you remember to place a semi-colon after the value (even if it is the last in the list).

Authorization Using the XUSER Key

`dbmcli -U <user_key> <command>`

Example: `dbmcli -U c db_state`

For information about XUSER see documentation *The SAP DB Database System →User Concept→User Data and XUSER*.

11 Documentation

11.1 Basic Information

Title	Explanation
The SAP DB Database System	Introduction to the architecture of the SAP DB database system, user and backup concepts, database tools, directory structure, database parameters, data management and processing, documentation, terminology, and so on.
Reference Manual: SAP DB 7.4	SQL statements and their syntax
SQL Mode ORACLE: SAP DB 7.4	Special features of SQL statements in the SQL mode ORACLE
System Tables: SAP DB 7.4	SAP DB system tables and their evaluation
Optimizer: SAP DB 7.4	Functions of the SAP DB Optimizer
Messages: SAP DB 7.4	List of all SAP DB messages

11.2 Tools

Title	Explanation
Database Manager	Creating and managing database instances
Database Manager GUI: SAP DB 7.4	Graphical user interface
Web DBM: SAP DB 7.4	Web-based user interface
Database Manager CLI: SAP DB 7.4	Command line user interface
External Backup Tools: SAP DB 7.4	Configuration and use of ADSM/TSM, Backint for Oracle, Backint for SAP DB and NetWorker
Query Tools	Data queries and data processing
SQL Studio: SAP DB 7.4	Graphical query tool

Web SQL Studio: SAP DB 7.4	Web-based query tool
Loader: SAP DB 7.4	Loading and unloading data
Database Analyzer: SAP DB	Analysis of the performance of database instances

11.3 Installation Documentation

Title	Explanation
Database Software Installation Guide: SAP DB 7.4	Standard installation, update, and uninstallation of all SAP DB software
Web Tools Installation Guide: SAP DB 7.4	Installation of SAP DB Web Tools
Installation Manual: SAP DB	Installation, update and uninstallation of installation profiles and individual SAP DB software components, updates of database instances, including software

11.4 Interfaces

Title	Explanation
C/C++ Precompiler User Manual: SAP DB 7.4	Options for the C/C++ Precompiler, Embedded SQL
ODBC Manual: SAP DB	Basics and special features of the SAP DB ODBC driver

11.5 Development

Title	Explanation
Development Environment: SAP DB	Usage of the development environment
Source Text for Tools: SAP DB	Creating source texts
Web Based Problem Tracking System: SAP DB	Web interface for internal SAP DB program PTS (Problem Tracking System) for documenting problem messages about SAP DB software

11.6 SAP DB Documentation

- For current SAP DB documentation and other important information, see the SAP DB Homepage at **`http://www.sapdb.org`**:
  Choose *Documentation* → *SAP DB Online Library* or *Download*.
- For SAP DB documentation for SAP systems, see the Help Portal at **`http://help.sap.com`**:
  Choose *SAP NetWeaver* → *SAP Web Application Server* → *SAP Web Application Server <release>* → *<language>* → *SAP NetWeaver Components* → *SAP DB*.
- For liveCache documentation for SAP systems, see the Help Portal at **`http://help.sap.com`**:
  Choose *SAP NetWeaver* → *SAP Web Application Server* → *SAP Web Application Server <release>* → *<language>* → *SAP NetWeaver Components* → *liveCache*.