

Db2 11.1 for Linux, UNIX, and Windows



Configuration Parameters

Db2 11.1 for Linux, UNIX, and Windows



Configuration Parameters

Notice regarding this document

This document in PDF form is provided as a courtesy to customers who have requested documentation in this format. It is provided As-Is without warranty or maintenance commitment.

Contents

Notice regarding this document iii

Figures ix

Tables xi

Configuration parameters 1

Configuring the Db2 database manager with configuration parameters	2
Configuration parameters summary	5
Configuration parameters that affect the number of agents	21
Configuration parameters that affect query optimization	22
Recompiling a query after configuration changes	24
Restrictions and behavior when configuring max_coordagents and max_connections	24
Failsafe option to update the database manager configuration file	26
Database manager configuration parameters	27
agent_stack_sz - Agent stack size	27
agentpri - Priority of agents	29
alt_diagpath - Alternate diagnostic data directory path	30
alternate_auth_enc - Alternate encryption algorithm for incoming connections at server configuration parameter	32
aslheapsz - Application support layer heap size	33
audit_buf_sz - Audit buffer size	34
authentication - Authentication type	35
cf_diaglevel - diagnostic error capture level configuration parameter for the CF	37
cf_diagpath - diagnostic data directory path configuration parameter for the CF	38
cf_mem_sz - CF memory configuration parameter	39
cf_num_conns - Number of CF connections per member per CF configuration parameter	39
cf_num_workers - Number of worker threads configuration parameter	40
cf_transport_method - Network transport method	41
catalog_noauth - Cataloging allowed without authority	42
clnt_krb_plugin - Client Kerberos plug-in	42
clnt_pw_plugin - Client userid-password plug-in	43
cluster_mgr - Cluster manager name	43
comm_bandwidth - Communications bandwidth	43
comm_exit_list - Communication exit library list	44
conn_elapse - Connection elapse time	45
cpuspeed - CPU speed	45
cur_eff_arch_level - Current effective architecture level configuration parameter	46
cur_eff_code_level - Current effective code level configuration parameter	46
date_compat - Date compatibility database configuration parameter	47

dft_account_str - Default charge-back account	47
dft_monswitches - Default database system monitor switches	48
dftdbpath - Default database path	49
diaglevel - Diagnostic error capture level	50
diagpath - Diagnostic data directory path	51
diagsize - Rotating diagnostic and administration notification logs configuration parameter	54
dir_cache - Directory cache support	56
discover - Discovery mode	57
discover_inst - Discover server instance	58
fcm_buffer_size - Inter-member buffer size	59
fcm_num_buffers - Number of FCM buffers	59
fcm_num_channels - Number of FCM channels	60
fcm_parallelism - Internode communication parallelism	61
fed_noauth - Bypass federated authentication	62
federated - Federated database system support	62
federated_async - Maximum asynchronous TQs per query configuration parameter	62
fenced_pool - Maximum number of fenced processes	63
group_plugin - Group plug-in	64
health_mon - Health monitoring	65
indexrec - Index re-creation time	65
instance_memory - Instance memory	68
intra_parallel - Enable intrapartition parallelism	71
java_heap_sz - Maximum Java interpreter heap size	72
jdk_path - Software Developer's Kit for Java installation path	72
keepfenced - Keep fenced process	73
encrypted_database - Database encryption state	74
keystore_location - Keystore location	74
keystore_type - Keystore type	75
local_gssplugin - GSS API plug-in used for local instance level authorization	76
max_connections - Maximum number of client connections	76
max_connretries - Node connection retries	77
max_coordagents - Maximum number of coordinating agents.	78
max_querydegree - Maximum query degree of parallelism	79
max_time_diff - Maximum time difference between members	80
maxagents - Maximum number of agents	80
maxcagents - Maximum number of concurrent agents	81
mon_heap_sz - Database system monitor heap size	82
nodetype - Instance node type	83
notifylevel - Notify level	84
num_initagents - Initial number of agents in pool	85
num_initfenced - Initial number of fenced processes	85

num_poolagents - Agent pool size	86	util_impact_lim - Instance impact policy	110
numdb - Maximum number of concurrently active databases including host and System i databases	87	wlm_dispatcher - Workload management dispatcher	111
query_heap_sz - Query heap size	88	wlm_disp_concur - Workload manager dispatcher thread concurrency	112
release - Configuration file release level	89	wlm_disp_cpu_shares - Workload manager dispatcher CPU shares	113
rstrt_light_mem - Restart light memory configuration parameter	89	wlm_disp_min_util - Workload manager dispatcher minimum CPU utilization	113
resync_interval - Transaction resync interval	90	Db2 database configuration parameters	114
rrioblk - Client I/O block size	91	alt_collate - Alternate collating sequence	114
sheaphthres - Sort heap threshold	92	app_ctl_heap_sz - Application control heap size	115
spm_log_file_sz - Sync point manager log file size	94	appgroup_mem_sz - Maximum size of application group memory set	116
spm_log_path - Sync point manager log file path	95	appl_memory - Application Memory configuration parameter	117
spm_max_resync - Sync point manager resync agent limit.	95	applheapsz - Application heap size	120
spm_name - Sync point manager name	96	archretrydelay - Archive retry delay on error	121
srvcon_auth - Authentication type for incoming connections at the server	96	auto_del_rec_obj - Automated deletion of recovery objects configuration parameter	121
srvcon_gssplugin_list - List of GSS API plug-ins for incoming connections at the server	97	auto_maint - Automatic maintenance	121
srvcon_pw_plugin - Userid-password plug-in for incoming connections at the server	97	auto_reval - Automatic revalidation and invalidation configuration parameter	123
srv_plugin_mode - Server plug-in mode.	98	autorestart - Auto restart enable	124
ssl_cipherspecs - Supported cipher specifications at the server configuration parameter.	98	avg_appls - Average number of active applications	125
ssl_clnt_keydb - SSL key file path for outbound SSL connections at the client configuration parameter	99	backup_pending - Backup pending indicator	126
ssl_clnt_stash - SSL stash file path for outbound SSL connections at the client configuration parameter	100	blk_log_dsk_ful - Block on log disk full	126
ssl_svr_keydb - SSL key file path for incoming SSL connections at the server configuration parameter	101	blocknonlogged - Block creation of tables that allow non-logged activity	127
ssl_svr_label - Label in the key file for incoming SSL connections at the server configuration parameter	101	CF self-tuning memory database configuration parameter	128
ssl_svr_stash - SSL stash file path for incoming SSL connections at the server configuration parameter	102	cf_catchup_trgt - Target for catch up time of secondary cluster caching facility configuration parameter	128
start_stop_time - Start and stop timeout	102	cf_db_mem_sz - Database memory configuration parameter	129
ssl_svcname - SSL service name configuration parameter	103	cf_gbp_sz - Group buffer pool configuration parameter	130
ssl_versions - Supported SSL versions at the server configuration parameter	104	cf_lock_sz - CF Lock manager configuration parameter	130
svcname - TCP/IP service name.	104	cf_sca_sz - Shared communication area configuration parameter	131
sysadm_group - System administration authority group name	105	catalogcache_sz - Catalog cache size.	132
sysctrl_group - System control authority group name	106	chnpggs_thresh - Changed pages threshold	133
sysmaint_group - System maintenance authority group name	106	codepage - Code page for the database.	134
sysmon_group - System monitor authority group name	106	codeset - Codeset for the database	134
tm_database - Transaction manager database name	107	collate_info - Collating information	134
tp_mon_name - Transaction processor monitor name	108	connect_proc - Connect procedure name database configuration parameter	135
trust_allclnts - Trust all clients.	109	country/region - Database territory code	136
trust_clntauth - Trusted clients authentication	110	cur_commit - Currently committed configuration parameter	136
		database_consistent - Database is consistent	137
		database_level - Database release level	138
		database_memory - Database shared memory size.	138
		dbheap - Database heap.	142
		db_mem_thresh - Database memory threshold	144
		date_compat - Date compatibility database configuration parameter	145

dec_arithmetic - DECIMAL arithmetic mode	146	logarchcompr1 - Primary archived log file compression configuration parameter	181
dec_to_char_fmt - Decimal to character function configuration parameter	147	logarchcompr2 - Secondary archived log file compression configuration parameter	181
decflt_rounding - Decimal floating point rounding configuration parameter	148	logarchmeth1 - Primary log archive method	182
dft_degree - Default degree.	150	logarchmeth2 - Secondary log archive method	183
dft_extent_sz - Default extent size of table spaces	151	logarchopt1 - Primary log archive options	185
dft_loadrec_ses - Default number of load recovery sessions	151	logarchopt2 - Secondary log archive options	185
dft_mttb_types - Default maintained table types for optimization	152	logbufsz - Log buffer size	186
dft_prefetch_sz - Default prefetch size	152	logfilsiz - Size of log files	187
dft_queryopt - Default query optimization class	153	loghead - First active log file	188
dft_refresh_age - Default refresh age.	154	logindexbuild - Log index pages created	188
dft_schemas_dcc - Default data capture on new schemas configuration parameter.	155	logpath - Location of log files	189
dft_sqlmathwarn - Continue upon arithmetic exceptions	155	logprimary - Number of primary log files	189
dft_table_org - Default table organization	157	logsecond - Number of secondary log files	190
discover_db - Discover database	157	max_log - Maximum log per transaction	192
dlchktime - Time interval for checking deadlock enable_xmlchar - Enable conversion to XML configuration parameter.	158 159	maxappls - Maximum number of active applications	193
encrlib - Encryption library.	159	maxfilop - Maximum database files open per database	194
encropts - Encryption options	160	maxlocks - Maximum percent of lock list before escalation.	195
extended_row_sz - Extended row size	160	min_dec_div_3 - Decimal division scale to 3	196
failarchpath - Failover log archive path.	161	mincommit - Number of commits to group	198
groupheap_ratio - Percent of memory for application group heap	161	mirrorlogpath - Mirror log path	199
hadr_db_role - HADR database role.	162	mon_act_metrics - Monitoring activity metrics configuration parameter.	201
hadr_local_host - HADR local host name	162	mon_deadlock - Monitoring deadlock configuration parameter.	202
hadr_local_svc - HADR local service name	163	mon_locktimeout - Monitoring lock timeout configuration parameter.	203
hadr_peer_window - HADR peer window configuration parameter.	163	mon_lockwait - Monitoring lock wait configuration parameter.	204
hadr_remote_host - HADR remote host name	165	mon_lw_thresh - Monitoring lock wait threshold configuration parameter.	205
hadr_remote_inst - HADR instance name of the remote server	165	mon_lck_msg_lvl - Monitoring lock event notification messages configuration parameter	205
hadr_remote_svc - HADR remote service name	166	mon_obj_metrics - Monitoring object metrics configuration parameter.	205
hadr_replay_delay - HADR replay delay configuration parameter.	167	mon_pkglst_sz - Monitoring package list size configuration parameter.	208
hadr_spool_limit - HADR log spool limit configuration parameter.	168	mon_req_metrics - Monitoring request metrics configuration parameter.	208
HADR_SSL_LABEL - Label name in the key file for SSL communication between HADR primary and standby instances configuration parameter	169	mon_rtn_data - Monitoring routine capture	209
hadr_syncmode - HADR synchronization mode for log writes in peer state	169	mon_rtn_execlist - Monitoring routine executable list	210
hadr_target_list - HADR target list database configuration parameter.	172	mon_uow_data - Monitoring unit of work events configuration parameter	211
hadr_timeout - HADR timeout value	174	mon_uow_execlist - Monitoring unit of work events with executable list configuration parameter	212
indexrec - Index re-creation time	174	mon_uow_pkglst - Monitoring unit of work events with package list configuration parameter	212
locklist - Maximum storage for lock list	176	multiplane_alloc - Multipage file allocation enabled	213
locktimeout - Lock timeout.	179	nchar_mapping - National character mapping	213
log_appl_info - Application information log record database configuration parameter	180	newlogpath - Change the database log path	214
log_ddl_stmts - Log Data Definition Language (DDL) statements database configuration parameter	180	num_db_backups - Number of database backups	215
log_retain_status - Log retain status indicator	180		

num_freqvalues - Number of frequent values retained	216
num_iocleaners - Number of asynchronous page cleaners	217
num_ioservers - Number of I/O servers	218
num_log_span - Number log span	219
num_quantiles - Number of quantiles for columns	220
numarchretry - Number of retries on error	221
numsegs - Default number of SMS containers	222
number_compat - Number compatibility database configuration parameter	222
opt_direct_wrkld - Optimize directed workload configuration parameter	222
overflowlogpath - Overflow log path	223
page_age_trgt_gcr - Page age target group crash recovery configuration parameter.	224
page_age_trgt_mcr - Page age target member crash recovery configuration parameter.	225
pagesize - Database default page size	225
pckcachesz - Package cache size	225
pl_stack_trace	227
priv_mem_thresh - Private memory threshold	228
rec_his_retentn - Recovery history retention period.	228
restore_pending - Restore pending	229
restrict_access - Database has restricted access configuration parameter.	229
rollfwd_pending - Roll forward pending indicator	230
section_actuals - Section actuals configuration parameter	230
self_tuning_mem- Self-tuning memory	231
seqdetect - Sequential detection and readahead flag.	232
sheaphres_shr - Sort heap threshold for shared sorts	233
smtp_server - SMTP server	237
softmax - Recovery range and soft checkpoint interval	237
sortheap - Sort heap size	239
sql_ccflags - Conditional compilation flags	243
stat_heap_sz - Statistics heap size.	244
stmt_conc - Statement concentrator configuration parameter	244
stmtheap - Statement heap size	246
string_units - Default string units.	247
suspend_io - Database I/O operations state configuration parameter	248
systeme_period_adj - Adjust temporal SYSTEM_TIME period database configuration parameter	248
territory - Database territory	249
trackmod - Track modified pages enable	250
tsm_mgmtclass - Tivoli Storage Manager management class	250
tsm_nodename - Tivoli Storage Manager node name	250

tsm_owner - Tivoli Storage Manager owner name	251
tsm_password - Tivoli Storage Manager password.	251
user_exit_status - User exit status indicator	252
util_heap_sz - Utility heap size	252
varchar2_compat - varchar2 compatibility database configuration parameter	254
vendoropt - Vendor options	254
wlm_admission_ctrl - WLM admission control	254
wlm_agent_load_trgt - WLM agent load target	255
wlm_collect_int - Workload management collection interval configuration parameter	255
wlm_cpu_limit - WLM CPU limit	256
wlm_cpu_shares - WLM CPU shares	256
wlm_cpu_share_mode - WLM CPU share mode	256
Db2 Administration Server (DAS) configuration parameters	257
authentication - Authentication type DAS	257
contact_host - Location of contact list	258
das_codepage - DAS code page	258
das_territory - DAS territory	259
dasadm_group - DAS administration authority group name	259
db2system - Name of the Db2 server system	260
discover - DAS discovery mode	260
exec_exp_task - Execute expired tasks	261
jdk_64_path - 64-Bit Software Developer's Kit for Java installation path DAS.	261
jdk_path - Software Developer's Kit for Java installation path DAS.	262
sched_enable - Scheduler mode	262
sched_userid - Scheduler user ID.	263
smtp_server - SMTP server.	263
toolscat_db - Tools catalog database	263
toolscat_inst - Tools catalog database instance	264
toolscat_schema - Tools catalog database schema	264
Ingest utility configuration parameters	265
commit_count - Commit count configuration parameter	265
commit_period - Commit period configuration parameter	266
num_flushers_per_partition - Number of flushers per database partition configuration parameter	266
num_formatters - Number of formatters configuration parameter.	267
pipe_timeout - Pipe timeout configuration parameter	267
retry_count - Retry count configuration parameter	267
retry_period - Retry period configuration parameter	268
shm_max_size - Maximum size of shared memory configuration parameter.	268

Index 271

Figures

1. Relationship between database objects and configuration files. 2
2. Synchronization modes for high availability disaster recovery (HADR) 171

Tables

1. Configurable Database Manager Configuration Parameters	6
2. Informational Database Manager Configuration Parameters	10
3. Configurable Database Configuration Parameters	11
4. Informational Database Configuration Parameters	18
5. DAS Configuration Parameters	19
6. Ingest Utility Configuration Parameters	20
7. Description of the configuration parameter section headings	20
8. Summary of required database manager configuration parameter settings for service class management by the Db2 WLM dispatcher.	113
9. Operating system support	119
10. Comparison of database memory size and overflow	140
11. Operating system support	141
12. Decimal floating point rounding modes	149
13. Metrics returned with EXTENDED option	206
14. Mapping between national character strings and data types	213

Configuration parameters

When you create a Db2® database instance or a database, a configuration file is created with default parameter values. You can modify these parameter values to improve performance and other characteristics of the instance or database.

The disk space and memory allocated by the database manager on the basis of default values of the parameters might be sufficient to meet your needs. In some situations, however, you might not be able to achieve maximum performance using these default values.

Configuration files contain parameters that define values such as the resources allocated to the Db2 database products and to individual databases, and the diagnostic level. There are two types of configuration files:

- The database manager configuration file for each Db2 instance
- The database configuration file for each individual database.

The *database manager configuration file* is created when a Db2 instance is created. The parameters it contains affect system resources at the instance level, independent of any one database that is part of that instance. Values for many of these parameters can be changed from the system default values to improve performance or increase capacity, depending on your system's configuration.

There is one database manager configuration file for each client installation as well. This file contains information about the client enabler for a specific workstation. A subset of the parameters available for a server are applicable to the client.

Database manager configuration parameters are stored in a file named `db2system`. This file is created when the instance of the database manager is created. In Linux and UNIX environments, this file can be found in the `sql1ib` subdirectory for the instance of the database manager. In Windows, the default location of this file varies from edition to edition of the Windows family of operating systems. You can verify the default directory on Windows, check the setting of the **DB2INSTPROF** registry variable using the command `db2set DB2INSTPROF`. You can also change the default instance directory by changing the **DB2INSTPROF** registry variable. If the **DB2INSTPROF** variable is set, the file is in the instance subdirectory of the directory specified by the **DB2INSTPROF** variable.

Other profile-registry variables that specify where run-time data files should go should query the value of **DB2INSTPROF**. This includes the following variables:

- **DB2CLIINIPATH**
- **diagpath**
- **spm_log_path**

All database configuration parameters are stored in a file named `SQLDBCONF`. You cannot directly edit these files. You can only change or view these files via a supplied API or by a tool which calls that API.

In a partitioned database environment, this file resides on a shared file system so that all database partition servers have access to the same file. The configuration of the database manager is the same on all database partition servers.

Most of the parameters either affect the amount of system resources that are allocated to a single instance of the database manager, or they configure the setup of the database manager and the different communications subsystems based on environmental considerations. In addition, there are other parameters that serve informative purposes only and cannot be changed. All of these parameters have global applicability independent of any single database stored under that instance of the database manager.

A *database configuration file* is created when a database is created, and resides where that database resides. There is one configuration file per database. Its parameters specify, among other things, the amount of resource to be allocated to that database. Values for many of the parameters can be changed to improve performance or increase capacity. Different changes might be required, depending on the type of activity in a specific database.

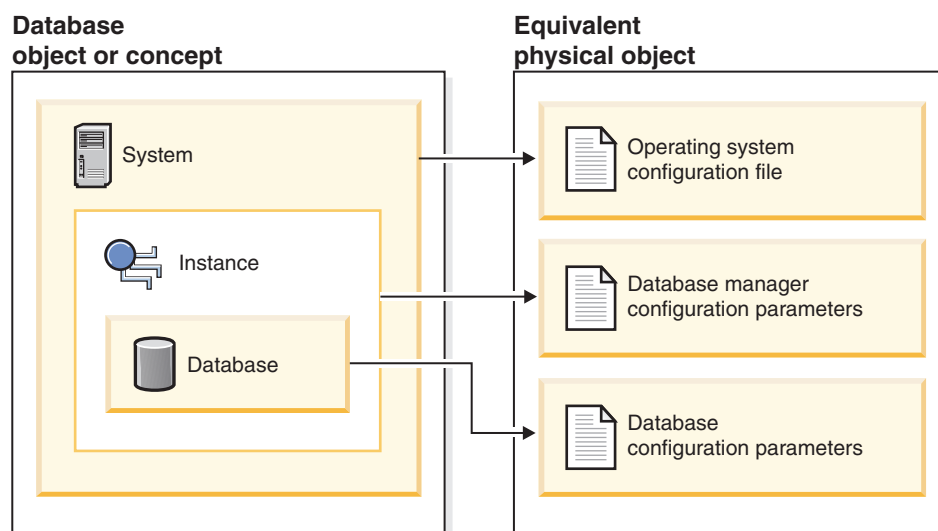


Figure 1. Relationship between database objects and configuration files

Configuring the Db2 database manager with configuration parameters

The disk space and memory that is allocated by the database manager based on default values of the parameters might be sufficient to meet your needs. However, in some situations you might not be able to achieve maximum performance by using these default values.

About this task

Since the default values are oriented towards workstations that have relatively small memory resources and are dedicated as database servers, you might need to modify these values if your environment has:

- Large databases
- Large numbers of connections
- High-performance requirements for a specific application
- Unique query or transaction loads or types

Each transaction processing environment is unique in one or more aspects. These differences can have a profound impact on the performance of the database

manager when you are using the default configuration. For this reason, you are advised to tune your configuration for your environment.

A good starting point for tuning your configuration is to use the Configuration Advisor or the **AUTOCONFIGURE** command. These tools generate values for parameters based on your responses to questions about workload characteristics.

Some configuration parameters can be set to **AUTOMATIC**, allowing the database manager to automatically manage these parameters to reflect the current resource requirements. To turn off the **AUTOMATIC** setting of a configuration parameter while you are maintaining the current internal setting, use the **MANUAL** keyword with the **UPDATE DATABASE CONFIGURATION** command. If the database manager updates the value of these parameters, the **GET DB CFG SHOW DETAIL** and **GET DBM CFG SHOW DETAIL** commands show the new value.

Parameters for an individual database are stored in a configuration file named **SQLDBCONF**. This file is stored along with other control files for the database in the **SQLnnnnn** directory, where **nnnnn** is a number that is assigned when the database was created. Each database has its own configuration file, and most of the parameters in the file specify the amount of resources that are allocated to that database. The file also contains descriptive information, and flags that indicate the status of the database.

If you edit **db2system**, **SQLDBCON**, or **SQLDBCONF** by using a method other than the methods provided by the database manager, you might make the database unusable. Do not change these files by using methods other than the methods documented and supported by the database manager.

In a partitioned database environment, a separate **SQLDBCONF** file exists for each database partition. The values in the **SQLDBCONF** file might be the same or different at each database partition. However, the recommendation is that in a homogeneous environment, the configuration parameter values must be the same on all database partitions. Typically, there might be a catalog node that needs different database configuration parameters setting, while the other data partitions have different values again, depending on their workstations types, and other information.

Procedure

1. Update configuration parameters.

- Using the command line processor

Commands to change the settings can be entered as follows:

For database manager configuration parameters:

- **GET DATABASE MANAGER CONFIGURATION** (or **GET DBM CFG**)
- **UPDATE DATABASE MANAGER CONFIGURATION** (or **UPDATE DBM CFG**)
- **RESET DATABASE MANAGER CONFIGURATION** (or **RESET DBM CFG**) to reset all database manager parameters to their default values
- **AUTOCONFIGURE**

For database configuration parameters:

- **GET DATABASE CONFIGURATION** (or **GET DB CFG**)
- **UPDATE DATABASE CONFIGURATION** (or **UPDATE DB CFG**)
- **RESET DATABASE CONFIGURATION** (or **RESET DB CFG**) to reset all database parameters to their default values
- **AUTOCONFIGURE**

- Using application programming interfaces (APIs)
The APIs can be called from an application or a host-language program. Call the following Db2 APIs to view or update configuration parameters:
 - db2AutoConfig - Access the Configuration Advisor
 - db2CfgGet - Get the database manager or database configuration parameters
 - db2CfgSet - Set the database manager or database configuration parameters
- Using common SQL application programming interface (API) procedures
You can call the common SQL API procedures from an SQL-based application, a Db2 command line, or a command script. Call the following procedures to view or update configuration parameters:
 - GET_CONFIG - Get the database manager or database configuration parameters
 - SET_CONFIG - Set the database manager or database configuration parameters
- Using IBM® Data Studio, right-click the instance to open the task assistant to update the database manager configuration parameters.

2. View updated configuration values.

For some database manager configuration parameters, the database manager must be stopped (**db2stop**) and then restarted (**db2start**) for the new parameter values to take effect.

For some database parameters, changes take effect only when the database is reactivated, or switched from offline to online. In these cases, all applications must first disconnect from the database. (If the database was activated, or switched from offline to online, then it must be deactivated and reactivated.) Then, at the first new connect to the database, the changes take effect.

If you change the setting of a configurable online database manager configuration parameter while you are attached to an instance, the default behavior of the **UPDATE DBM CFG** command is to apply the change immediately. If you do not want the change to be applied immediately, use the **DEFERRED** option on the **UPDATE DBM CFG** command.

To change a database manager configuration parameter online, use the following commands:

```
db2 attach to instance-name
db2 update dbm cfg using parameter-name value
db2 detach
```

For clients, changes to the database manager configuration parameters take effect the next time the client connects to a server.

If you change a configurable online database configuration parameter while connected, the default behavior is to apply the change online, wherever possible. Some parameter changes might take a noticeable amount of time to take effect due to the additional processing time associated with allocating space. To change configuration parameters online from the command line processor, a connection to the database is needed. To change a database configuration parameter online, use the following commands:

```
db2 connect to dbname
db2 update db cfg using parameter-name parameter-value
db2 connect reset
```

Each configurable online configuration parameter has a *propagation class* that is associated with it. The propagation class indicates when you can expect a change to the configuration parameter to take effect. The following is a list of four propagation classes:

- Immediate - Parameters that change immediately upon command or API invocation. For instance, **diaglevel** has a propagation class of immediate.
- Statement boundary - Parameters that change on statement and statement-like boundaries. For instance, if you change the value of **sorthheap**, all new requests use the new value.
- Transaction boundary - Parameters that change on transaction boundaries. For instance, a new value for **dl_expint** is updated after a COMMIT statement.
- Connection - Parameters that change on new connection to the database. For instance, a new value for **dft_degree** takes effect for new applications that are connecting to the database.

While new parameter values might not be immediately effective, viewing the parameter settings (by using the **GET DATABASE MANAGER CONFIGURATION** or **GET DATABASE CONFIGURATION** command) always shows the latest updates. Viewing the parameter settings by using the **SHOW DETAIL** clause on these commands shows both the latest updates and the values in memory.

3. Rebind applications after you update the database configuration parameters.

Changing some database configuration parameters can influence the access plan that is chosen by the SQL and XQuery optimizer. After you change any of these parameters, consider rebinding your applications to ensure that the best access plan is being used for your SQL and XQuery statements. Any parameters that were modified online (for example, by using the **UPDATE DATABASE CONFIGURATION IMMEDIATE** command) cause the SQL and XQuery optimizer to choose new access plans for new query statements. However, the query statement cache is not purged of existing entries. To clear the contents of the query cache, use the **FLUSH PACKAGE CACHE** statement.

Note: A number of configuration parameters (for example, **health_mon**) are described as having acceptable values of either Yes or No, or On or Off in the help and other Db2 documentation. To clarify, Yes must be considered equivalent to On and No must be considered equivalent to Off.

Configuration parameters summary

The following tables list the parameters in the database manager and database configuration files for database servers. When you change the database manager and database configuration parameters, consider the detailed information for each parameter. Specific operating environment information (including defaults) is part of each parameter description.

Database Manager Configuration Parameter Summary

For some database manager configuration parameters, the database manager must be stopped (**db2stop**) and restarted (**db2start**) for the new parameter values to take effect. Other parameters can be changed online; these parameters are called *configurable online configuration parameters*. If you change the setting of a configurable online database manager configuration parameter while you are attached to an instance, the default behavior of the **UPDATE DBM CFG** command applies the change immediately. If you do not want the change to be applied immediately, use the **DEFERRED** option on the **UPDATE DBM CFG** command.

The column “Auto” in the following table indicates whether the parameter supports the **AUTOMATIC** keyword on the **UPDATE DBM CFG** command.

When you update a parameter to automatic, it is also possible to specify a starting value and the **AUTOMATIC** keyword. The value can mean something different for each parameter, and in some cases it is not applicable. Before you specify a value, read the parameter's documentation to determine what it represents. In the following example, **num_poolagents** is updated to **AUTOMATIC** and the database manager uses 20 as the minimum number of idle agents to pool.

```
db2 update dbm cfg using num_poolagents 20 automatic
```

To unset the **AUTOMATIC** feature, the parameter can be updated to a value or the **MANUAL** keyword can be used. When a parameter is updated to **MANUAL**, the parameter is no longer automatic and is set to its current value (as displayed in the Current Value column from the **GET DBM CFG SHOW DETAIL** and **GET DB CFG SHOW DETAIL** commands).

If a database is created by either the **CREATE DATABASE**, or the **sqlecrea** API, then the Configuration Advisor runs by default to update the database configuration parameters with automatically computed values. If a database is created by either the **CREATE DATABASE** command with the **AUTOCONFIGURE APPLY NONE** clause added, or the **sqlecrea** API specifies not to run the Configuration Advisor, then the configuration parameters are set to the default values.

The column “Perf. Impact” provides an indication of the relative importance of each parameter as it relates to system performance. It is impossible for this column to apply accurately to all environments. You must view this information as a generalization.

- High — Indicates that the parameter can have a significant impact on performance. You must consciously decide the values of these parameters, which, in some cases, means that you accept the default values provided.
- Medium — Indicates that the parameter can have some impact on performance. Your specific environment and requirements determine how much tuning effort needs to be focused on these parameters.
- Low — Indicates that the parameter has a less general or less significant impact on performance.
- None — Indicates that the parameter does not directly impact performance. Although you do not have to tune these parameters for performance enhancement, they can be important for other aspects of your system configuration, such as communication support.

The columns “Token”, “Token Value”, and “Data Type” provide information that you need when you are calling the **db2CfgGet** or the **db2CfgSet** API. This information includes configuration parameter identifiers, entries for the *token* element in the **db2CfgParam** data structure, and data types for values that are passed to the structure.

Table 1. Configurable Database Manager Configuration Parameters

Parameter	Cfg. Online	Auto.	Perf. Impact	Token	Token Value	Data Type	Additional Information
agent_stack_sz	No	No	Low	SQLF_KTN_AGENT_STACK_SZ	61	UInt16	“agent_stack_sz - Agent stack size” on page 27
agentpri	No	No	High	SQLF_KTN_AGENTPRI	26	Sint16	“agentpri - Priority of agents” on page 29
alt_diagpath	Yes	No	None	SQLF_KTN_ALT_DIAGPATH SQLF_KTN_ALT_DIAGPATH_FULL	941	char [] (String)	“alt_diagpath - Alternate diagnostic data directory path” on page 30

Table 1. Configurable Database Manager Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Perf. Impact	Token	Token Value	Data Type	Additional Information
alternate_auth_enc ⁶	No	No	Low	SQLF_KTN_ALTERNATE_AUTH_ENC	938	Uint16	"alternate_auth_enc - Alternate encryption algorithm for incoming connections at server configuration parameter" on page 32
aslheapsz	No	No	High	SQLF_KTN_ASLHEAPSZ	15	Uint32	"aslheapsz - Application support layer heap size" on page 33
audit_buf_sz	No	No	High	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32	"audit_buf_sz - Audit buffer size" on page 34
authentication	No	No	Low	SQLF_KTN_AUTHENTICATION	78	Uint16	"authentication - Authentication type" on page 35
catalog_noauth	Yes	No	None	SQLF_KTN_CATALOG_NOAUTH	314	Uint16	"catalog_noauth - Cataloging allowed without authority" on page 42
cf_diaglevel	No	No	None	SQLF_KTN_CF_DIAGLEVEL	968	Uint16	"cf_diaglevel - diagnostic error capture level configuration parameter for the CF" on page 37
cf_diagpath	No	No	None	SQLF_KTN_CF_DIAGPATH SQLF_KTN_CF_DIAGPATH_FULL	969	char(215)	"cf_diagpath - diagnostic data directory path configuration parameter for the CF" on page 38
cf_mem_sz	No	Yes	High	SQLF_KTN_CF_MEM_SZ	960	Uint32	"cf_mem_sz - CF memory configuration parameter" on page 39
cf_num_conns	Yes	Yes	High	SQLF_KTN_CF_NUM_CONNS	966	Uint32	"cf_num_conns - Number of CF connections per member per CF configuration parameter" on page 39
cf_num_workers	No	Yes	High	SQLF_KTN_CF_NUM_WORKERS	961	Uint32	"cf_num_workers - Number of worker threads configuration parameter" on page 40
cf_transport_method	No	Yes	High	SQLF_KTN_CF_TRANSPORT_METHOD	10126	Uint16	"cf_transport_method - Network transport method" on page 41
clnt_krb_plugin	No	No	None	SQLF_KTN_CLNT_KRB_PLUGIN	812	char(33)	"clnt_krb_plugin - Client Kerberos plug-in" on page 42
clnt_pw_plugin	No	No	None	SQLF_KTN_CLNT_PW_PLUGIN	811	char(33)	"clnt_pw_plugin - Client userid-password plug-in" on page 43
cluster_mgr	No	No	None	SQLF_KTN_CLUSTER_MGR	920	char(262)	"cluster_mgr - Cluster manager name" on page 43
comm_bandwidth	Yes	No	Medium	SQLF_KTN_COMM_BANDWIDTH	307	float	"comm_bandwidth - Communications bandwidth" on page 43
comm_exit_list	No	No	Low	SQLF_KTN_COMM_EXIT_LIST	10121	char(129)	"comm_exit_list - Communication exit library list" on page 44
conn_elapse	Yes	No	Medium	SQLF_KTN_CONN_ELAPSE	508	Uint16	"conn_elapse - Connection elapse time" on page 45
cpuspeed	Yes	No	High	SQLF_KTN_CPUSPEED	42	float	"cpuspeed - CPU speed" on page 45
dft_account_str	Yes	No	None	SQLF_KTN_DFT_ACCOUNT_STR	28	char(25)	"dft_account_str - Default charge-back account" on page 47
dft_monswitches • dft_mon_bufpool • dft_mon_lock • dft_mon_sort • dft_mon_stmt • dft_mon_table • dft_mon_timestamp • dft_mon_uow	Yes	No	Medium	SQLF_KTN_DFT_MONSWITCHES ² • SQLF_KTN_DFT_MON_BUFPOOL • SQLF_KTN_DFT_MON_LOCK • SQLF_KTN_DFT_MON_SORT • SQLF_KTN_DFT_MON_STMT • SQLF_KTN_DFT_MON_TABLE • SQLF_KTN_DFT_MON_TIMESTAMP • SQLF_KTN_DFT_MON_UOW	29 • 33 • 34 • 35 • 31 • 32 • 36 • 30	Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16	"dft_monswitches - Default database system monitor switches" on page 48
dftdbpath	Yes	No	None	SQLF_KTN_DFTDBPATH	27	char(215)	"dftdbpath - Default database path" on page 49
diaglevel	Yes	No	Low	SQLF_KTN_DIAGLEVEL	64	Uint16	"diaglevel - Diagnostic error capture level" on page 50
diagpath	Yes	No	None	SQLF_KTN_DIAGPATH SQLF_KTN_DIAGPATH_FULL	65	char(215)	"diagpath - Diagnostic data directory path" on page 51
diagsz	No	No	Medium	SQLF_KTN_DIAGSZ	939	Uint64	"diagsz - Rotating diagnostic and administration notification logs configuration parameter" on page 54
dir_cache	No	No	Medium	SQLF_KTN_DIR_CACHE	40	Uint16	"dir_cache - Directory cache support" on page 56
discover ³	No	No	Medium	SQLF_KTN_DISCOVER	304	Uint16	"discover - Discovery mode" on page 57
discover_inst	Yes	No	Low	SQLF_KTN_DISCOVER_INST	308	Uint16	"discover_inst - Discover server instance" on page 58
fcm_buffer_size	No	No	Medium	SQLF_KTN_FCM_BUFFER_SIZE	10154	Sint32	"fcm_buffer_size - Inter-member buffer size" on page 59
fcm_num_buffers	Yes	Yes	Medium	SQLF_KTN_FCM_NUM_BUFFERS	503	Uint32	"fcm_num_buffers - Number of FCM buffers" on page 59
fcm_num_channels	Yes	Yes	Medium	SQLF_KTN_FCM_NUM_CHANNELS	902	Uint32	"fcm_num_channels - Number of FCM channels" on page 60
fcm_parallelism	No	No	High	SQLF_KTN_FCM_NUM_PARALLELISM	848	Sint32	"fcm_parallelism - Internode communication parallelism" on page 61
fed_noauth	Yes	No	None	SQLF_KTN_FED_NOAUTH	806	Uint16	"fed_noauth - Bypass federated authentication" on page 62

Table 1. Configurable Database Manager Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Perf. Impact	Token	Token Value	Data Type	Additional Information
federated	No	No	Medium	SQLF_KTN_FEDERATED	604	Sint16	"federated - Federated database system support" on page 62
federated_async	Yes	Yes	Medium	SQLF_KTN_FEDERATED_ASYNC	849	Sint32	"federated_async - Maximum asynchronous TQs per query configuration parameter" on page 62
fenced_pool	Yes	Yes	Medium	SQLF_KTN_FENCED_POOL	80	Sint32	"fenced_pool - Maximum number of fenced processes" on page 63
group_plugin	No	No	None	SQLF_KTN_GROUP_PLUGIN	810	char(33)	"group_plugin - Group plug-in" on page 64
health_mon	Yes	No	Low	SQLF_KTN_HEALTH_MON	804	Uint16	"health_mon - Health monitoring" on page 65
indexrec⁴	Yes	No	Medium	SQLF_KTN_INDEXREC	20	Uint16	"indexrec - Index re-creation time" on page 65
instance_memory	Yes	Yes	Medium	SQLF_KTN_INSTANCE_MEMORY	803	Uint64	"instance_memory - Instance memory" on page 68
intra_parallel	No	No	High	SQLF_KTN_INTRA_PARALLEL	306	Sint16	"intra_parallel - Enable intrapartition parallelism" on page 71
java_heap_sz	No	No	High	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32	"java_heap_sz - Maximum Java interpreter heap size" on page 72
jdk_path	No	No	None	SQLF_KTN_JDK_PATH	311	char(255)	"jdk_path - Software Developer's Kit for Java installation path" on page 72
keepfenced	No	No	Medium	SQLF_KTN_KEEFENCED	81	Uint16	"keepfenced - Keep fenced process" on page 73
local_gssplugin	No	No	None	SQLF_KTN_LOCAL_GSSPLUGIN	816	char(33)	"local_gssplugin - GSS API plug-in used for local instance level authorization" on page 76
max_connections	Yes	Yes	Medium	SQLF_KTN_MAX_CONNECTIONS	802	Sint32	"max_connections - Maximum number of client connections" on page 76
max_connretries	Yes	No	Medium	SQLF_KTN_MAX_CONNRETRIES	509	Uint16	"max_connretries - Node connection retries" on page 77
max_coordagents	Yes	Yes	Medium	SQLF_KTN_MAX_COORDAGENTS	501	Sint32	"max_coordagents - Maximum number of coordinating agents" on page 78
max_querydegree	Yes	No	High	SQLF_KTN_MAX_QUERYDEGREE	303	Sint32	"max_querydegree - Maximum query degree of parallelism" on page 79
max_time_diff	No	No	Medium	SQLF_KTN_MAX_TIME_DIFF	510	Uint16	"max_time_diff - Maximum time difference between members" on page 80
mon_heap_sz	Yes	Yes	Low	SQLF_KTN_MONHEAP_SZ	10156	Uint64	"mon_heap_sz - Database system monitor heap size" on page 82
notifylevel	Yes	No	Low	SQLF_KTN_NOTIFYLEVEL	605	Sint16	"notifylevel - Notify level" on page 84
num_initagents	No	No	Medium	SQLF_KTN_NUM_INITAGENTS	500	Uint32	"num_initagents - Initial number of agents in pool" on page 85
num_initfenced	No	No	Medium	SQLF_KTN_NUM_INITFENCED	601	Sint32	"num_initfenced - Initial number of fenced processes" on page 85
num_poolagents	Yes	Yes	High	SQLF_KTN_NUM_POOLAGENTS	502	Sint32	"num_poolagents - Agent pool size" on page 86
numdb	No	No	Low	SQLF_KTN_NUMDB	6	Uint16	"numdb - Maximum number of concurrently active databases including host and System i [®] databases" on page 87
query_heap_sz	No	No	Medium	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32	"query_heap_sz - Query heap size" on page 88
rstrt_light_mem	No	Yes	Medium	SQLF_KTN_RSTRT_LIGHT_MEM	967	Uint16	"rstrt_light_mem - Restart light memory configuration parameter" on page 89
resync_interval	No	No	None	SQLF_KTN_RESYNC_INTERVAL	68	Uint16	"resync_interval - Transaction resync interval" on page 90
rqrioblk	No	No	High	SQLF_KTN_RQRIOBLK	1	Uint16	"rqrioblk - Client I/O block size" on page 91
sheapthres	No	No	High	SQLF_KTN_SHEAPTHRES	21	Uint32	"sheapthres - Sort heap threshold" on page 92
spm_log_file_sz	No	No	Low	SQLF_KTN_SPM_LOG_FILE_SZ	90	Sint32	"spm_log_file_sz - Sync point manager log file size" on page 94
spm_log_path	No	No	Medium	SQLF_KTN_SPM_LOG_PATH	313	char(226)	"spm_log_path - Sync point manager log file path" on page 95
spm_max_resync	No	No	Low	SQLF_KTN_SPM_MAX_RESYNC	91	Sint32	"spm_max_resync - Sync point manager resync agent limit" on page 95
spm_name	No	No	None	SQLF_KTN_SPM_NAME	92	char(8)	"spm_name - Sync point manager name" on page 96
srvcon_auth	No	No	None	SQLF_KTN_SRVCON_AUTH	815	Uint16	"srvcon_auth - Authentication type for incoming connections at the server" on page 96
srvcon_gssplugin_list	No	No	None	SQLF_KTN_SRVCON_GSSPLUGIN_LIST	814	char(256)	"srvcon_gssplugin_list - List of GSS API plug-ins for incoming connections at the server" on page 97
srv_plugin_mode	No	No	None	SQLF_KTN_SRV_PLUGIN_MODE	809	Uint16	"srv_plugin_mode - Server plug-in mode" on page 98
srvcon_pw_plugin	No	No	None	SQLF_KTN_SRVCON_PW_PLUGIN	813	char(33)	"srvcon_pw_plugin - Userid-password plug-in for incoming connections at the server" on page 97

Table 1. Configurable Database Manager Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Perf. Impact	Token	Token Value	Data Type	Additional Information
ssl_svr_keydb	No	No	None	SQLF_KTN_SSL_SVR_KEYDB	930	char(1023)	"ssl_svr_keydb - SSL key file path for incoming SSL connections at the server configuration parameter" on page 101
ssl_svr_stash	No	No	None	SQLF_KTN_SSL_SVR_STASH	931	char(1023)	"ssl_svr_stash - SSL stash file path for incoming SSL connections at the server configuration parameter" on page 102
ssl_svr_label	No	No	None	SQLF_KTN_SSL_SVR_LABEL	932	char(1023)	"ssl_svr_label - Label in the key file for incoming SSL connections at the server configuration parameter" on page 101
ssl_svcname	No	No	None	SQLF_KTN_SSL_SVCNAME	933	char(14)	"ssl_svcname - SSL service name configuration parameter" on page 103
ssl_cipherspecs	No	No	None	SQLF_KTN_SSL_CIPHERSPEC	934	char(255)	"ssl_cipherspecs - Supported cipher specifications at the server configuration parameter" on page 98
ssl_versions	No	No	None	SQLF_KTN_SSL_VERSIONS	935	char(255)	"ssl_versions - Supported SSL versions at the server configuration parameter" on page 104
ssl_clnt_keydb	No	No	None	SQLF_KTN_SSL_CLNT_KEYDB	936	char(1023)	"ssl_clnt_keydb - SSL key file path for outbound SSL connections at the client configuration parameter" on page 99
ssl_clnt_stash	No	No	None	SQLF_KTN_SSL_CLNT_STASH	937	char(1023)	"ssl_clnt_stash - SSL stash file path for outbound SSL connections at the client configuration parameter" on page 100
start_stop_time	Yes	No	Low	SQLF_KTN_START_STOP_TIME	511	UInt16	"start_stop_time - Start and stop timeout" on page 102
svcname	No	No	None	SQLF_KTN_SVCNAME	24	char(14)	"svcname - TCP/IP service name" on page 104
sysadm_group	No	No	None	SQLF_KTN_SYSADM_GROUP	39	char(128)	"sysadm_group - System administration authority group name" on page 105
sysctrl_group	No	No	None	SQLF_KTN_SYSCTRL_GROUP	63	char(128)	"sysctrl_group - System control authority group name" on page 106
sysmaint_group	No	No	None	SQLF_KTN_SYSMAINT_GROUP	62	char(128)	"sysmaint_group - System maintenance authority group name" on page 106
sysmon_group	No	No	None	SQLF_KTN_SYSMON_GROUP	808	char(128)	"sysmon_group - System monitor authority group name" on page 106
tm_database	No	No	None	SQLF_KTN_TM_DATABASE	67	char(8)	"tm_database - Transaction manager database name" on page 107
tp_mon_name	No	No	None	SQLF_KTN_TP_MON_NAME	66	char(19)	"tp_mon_name - Transaction processor monitor name" on page 108
trust_allclnts ⁵	No	No	None	SQLF_KTN_TRUST_ALLCLNTS	301	UInt16	"trust_allclnts - Trust all clients" on page 109
trust_clntauth	No	No	None	SQLF_KTN_TRUST_CLNTAUTH	302	UInt16	"trust_clntauth - Trusted clients authentication" on page 110
util_impact_lim	Yes	No	High	SQLF_KTN_UTIL_IMPACT_LIM	807	UInt32	"util_impact_lim - Instance impact policy" on page 110
wlm_dispatcher	Yes	No	Medium	SQLF_KTN_WLM_DISPATCHER	976	UInt16	"wlm_dispatcher - Workload management dispatcher" on page 111
wlm_disp_concur	Yes	No	Low	SQLF_KTN_WLM_DISP_CONCUR	977	Sint16	"wlm_disp_concur - Workload manager dispatcher thread concurrency" on page 112
wlm_disp_cpu_shares	Yes	No	Low	SQLF_KTN_WLM_DISP_CPU_SHARES	979	UInt16	"wlm_disp_cpu_shares - Workload manager dispatcher CPU shares" on page 113
wlm_disp_min_util	Yes	No	Low	SQLF_KTN_WLM_DISP_MIN_UTIL	978	UInt16	"wlm_disp_min_util - Workload manager dispatcher minimum CPU utilization" on page 113

Table 1. Configurable Database Manager Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Perf. Impact	Token	Token Value	Data Type	Additional Information
<p>Note:</p> <p>Refer to the header files <code>sqlenv.h</code> and <code>sqlutil.h</code> for the valid values and for the definitions that are used by the configuration parameters.</p> <p>1.</p> <ul style="list-style-type: none"> Bit 1 (xxxx xxx1): <code>dft_mon_uow</code> Bit 2 (xxxx xx1x): <code>dft_mon_stmt</code> Bit 3 (xxxx x1xx): <code>dft_mon_table</code> Bit 4 (xxxx 1xxx): <code>dft_mon_buffpool</code> Bit 5 (xx1 xxxx): <code>dft_mon_lock</code> Bit 6 (xx1x xxxx): <code>dft_mon_sort</code> Bit 7 (x1xx xxxx): <code>dft_mon_timestamp</code> <p>2. Valid values</p> <ul style="list-style-type: none"> <code>SQLF_DSCVR_KNOWN (1)</code> <code>SQLF_DSCVR_SEARCH (2)</code> <p>3. Valid values</p> <ul style="list-style-type: none"> <code>SQLF_INX_REC_SYSTEM (0)</code> <code>SQLF_INX_REC_REFERENCE (1)</code> <code>SQLF_INX_REC_RESTART (2)</code> <code>SQLF_INX_REC_RESTART_NO_REDO (3)</code> <code>SQLF_INX_REC_ACCESS_NO_REDO (4)</code> <p>4. Valid values</p> <ul style="list-style-type: none"> <code>SQLF_TRUST_ALLCLNTS_NO (0)</code> <code>SQLF_TRUST_ALLCLNTS_YES (1)</code> <code>SQLF_TRUST_ALLCLNTS_DRDAONLY (2)</code> <p>5. Valid values</p> <ul style="list-style-type: none"> <code>SQL_ALTERNATE_AUTH_ENC_AES (0)</code> <code>SQL_ALTERNATE_AUTH_ENC_AES_CMP (1)</code> <code>SQL_ALTERNATE_AUTH_ENC_NOTSPEC (255)</code> 							

Table 2. Informational Database Manager Configuration Parameters

Parameter	Token	Token Value	Data Type	Additional Information
<code>curr_eff_arch_level</code>	<code>SQLF_KTN_CURR_EFF_ARCH_LVL</code>	10116	UInt64	" <code>cur_eff_arch_level</code> - Current effective architecture level configuration parameter" on page 46
<code>curr_eff_code_level</code>	<code>SQLF_KTN_CURR_EFF_CODE_LVL</code>	10120	UInt64	" <code>cur_eff_code_level</code> - Current effective code level configuration parameter" on page 46
<code>nodetype</code> ¹	<code>SQLF_KTN_NODETYPE</code>	100	UInt16	" <code>nodetype</code> - Instance node type" on page 83
<code>release</code>	<code>SQLF_KTN_RELEASE</code>	101	UInt16	" <code>release</code> - Configuration file release level" on page 89
<p>Note:</p> <p>Refer to the header files <code>sqlenv.h</code> and <code>sqlutil.h</code> for the valid values and for the definitions that are used by the configuration parameters.</p> <p>1. Valid values</p> <ul style="list-style-type: none"> <code>SQLF_NT_STANDALONE (0)</code> <code>SQLF_NT_SERVER (1)</code> <code>SQLF_NT_REQUESTOR (2)</code> <code>SQLF_NT_STAND_REQ (3)</code> <code>SQLF_NT_MPP (4)</code> <code>SQLF_NT_SATELLITE (5)</code> 				

Database Configuration Parameter Summary

The following table lists the parameters in the database configuration file. When you are changing the database configuration parameters, consider the detailed information for the parameter.

For some database configuration parameters, changes take effect when the database is reactivated. In these cases, all applications must first disconnect from the database. (If the database was activated, then it must be deactivated and reactivated.) The changes take effect at the next connection to the database. Other parameters can be changed online; these parameters are called *configurable online configuration parameters*.

Refer to the Database Manager Configuration Parameter Summary section for a description of the “Auto.”, “Perf. Impact”, “Token”, “Token Value”, and “Data Type” columns.

The column “Member Cfg.” applies only for a Db2 pureScale® environment. While all database configuration parameters can be configured globally, the column indicates whether a database configuration parameter is able to be configured on a per-member basis. For more information about the database configuration parameters that are configurable on a per-member basis, see Db2 pureScale Feature database configuration parameters.

The **AUTOMATIC** keyword is also supported on the **UPDATE DB CFG** command. In the following example, **database_memory** is updated to **AUTOMATIC** and the database manager uses 20000 as a starting value when you are making further changes to this parameter.

```
db2 update db cfg using for sample using database_memory 20000 automatic
```

Starting with Version 9.5, you can update and reset database configuration parameter values across some or all partitions without having to issue the **db2_a11** command. Furthermore, you do not have to update or reset each partition individually.

If a database is created by either the **CREATE DATABASE**, or the **sqlecrea** API, then the Configuration Advisor runs by default to update the database configuration parameters with automatically computed values. If a database is created by either the **CREATE DATABASE** command with the **AUTOCONFIGURE APPLY NONE** clause added, or the **sqlecrea** API specifies not to run the Configuration Advisor, then the configuration parameters are set to the default values.

Table 3. Configurable Database Configuration Parameters

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
alt_collate	No	No	No	None	SQLF_DBTN_ALT_COLLATE	809	UInt32	“alt_collate - Alternate collating sequence” on page 114
applheapsz	Yes	Yes	Yes	Medium	SQLF_DBTN_APPLHEAP_SZ	10157	UInt64	“applheapsz - Application heap size” on page 120
appl_memory	Yes	Yes	Yes	Medium	SQLF_DBTN_APPL_MEMORY	904	UInt64	“appl_memory - Application Memory configuration parameter” on page 117
archretrydelay	Yes	No	No	None	SQLF_DBTN_ARCHRETRYDELAY	828	UInt16	“archretrydelay - Archive retry delay on error” on page 121
<ul style="list-style-type: none"> • auto_maint • auto_db_backup • auto_tbl_maint • auto_runstats • auto_stmt_stats • auto_stats_views • auto_reorg 	Yes	No	No	Medium	<ul style="list-style-type: none"> • SQLF_DBTN_AUTO_MAINT • SQLF_DBTN_AUTO_DB_BACKUP • SQLF_DBTN_AUTO_TBL_MAINT • SQLF_DBTN_AUTO_RUNSTATS • SQLF_DBTN_AUTO_STMT_STATS • SQLF_DBTN_AUTO_STATS_VIEWS • SQLF_DBTN_AUTO_REORG 	<ul style="list-style-type: none"> • 831 • 833 • 835 • 837 • 905 • 980 • 841 	UInt32	“auto_maint - Automatic maintenance” on page 121

Table 3. Configurable Database Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
auto_del_rec_obj	Yes	No	No	Medium	SQLF_DBTN_AUTO_DEL_REC_OBJ	912	UInt16	"auto_del_rec_obj - Automated deletion of recovery objects configuration parameter" on page 121
autorestart	Yes	No	No	Low	SQLF_DBTN_AUTO_RESTART	25	UInt16	"autorestart - Auto restart enable" on page 124
auto_reval	Yes	No	Yes	Medium	SQLF_DBTN_AUTO_REVAL	920	UInt16	"auto_reval - Automatic revalidation and invalidation configuration parameter" on page 123
avg_appls	Yes	Yes	Yes	High	SQLF_DBTN_AVG_APPLS	47	UInt16	"avg_appls - Average number of active applications" on page 125
blk_log_dsk_ful	Yes	No	Yes	None	SQLF_DBTN_BLK_LOG_DSK_FUL	804	UInt16	"blk_log_dsk_ful - Block on log disk full" on page 126
blocknonlogged	Yes	No	Yes	Low	SQLF_DBTN_BLOCKNONLOGGED	940	UInt16	"blocknonlogged - Block creation of tables that allow non-logged activity" on page 127
catalogcache_sz	Yes	No	Yes	Medium	SQLF_DBTN_CATALOGCACHE_SZ	56	UInt32	"catalogcache_sz - Catalog cache size" on page 132
cf_catchup_trgt	Yes	Yes	No	High	SQLF_DBTN_CF_CATCHUP_TRGT	970	UInt16	"cf_catchup_trgt - Target for catch up time of secondary cluster caching facility configuration parameter" on page 128
cf_db_mem_sz	Yes	Yes	No	Low	SQLF_DBTN_CF_DB_MEM_SZ	923	UInt32	"cf_db_mem_sz - Database memory configuration parameter" on page 129
cf_gbp_sz	Yes	Yes	No	High	SQLF_DBTN_CF_GBP_SZ	920	UInt32	"cf_gbp_sz - Group buffer pool configuration parameter" on page 130
cf_lock_sz	Yes	Yes	No	High	SQLF_DBTN_CF_LOCK_SZ	921	UInt32	"cf_lock_sz - CF Lock manager configuration parameter" on page 130
cf_sca_sz	Yes	Yes	No	High	SQLF_DBTN_CF_SCA_SZ	922	UInt32	"cf_sca_sz - Shared communication area configuration parameter" on page 131
chngpgs_thresh	No	No	Yes	High	SQLF_DBTN_CHNGPGS_THRESH	38	UInt16	"chngpgs_thresh - Changed pages threshold" on page 133
connect_proc	Yes	No	No	None	SQLF_DBTN_CONNECT_PROC	954	char(257)	"connect_proc - Connect procedure name database configuration parameter" on page 135
cur_commit	No	No	Yes	Medium	SQLF_DBTN_CUR_COMMIT	917	UInt32	"cur_commit - Currently committed configuration parameter" on page 136
database_memory	Yes	Yes	Yes	Medium	SQLF_DBTN_DATABASE_MEMORY	803	UInt64	"database_memory - Database shared memory size" on page 138
dbheap	Yes	Yes	Yes	Medium	SQLF_DBTN_DB_HEAP	58	UInt64	"dbheap - Database heap" on page 142
db_mem_thresh	Yes	No	Yes	Medium	SQLF_DBTN_DB_MEM_THRESH	849	UInt16	"db_mem_thresh - Database memory threshold" on page 144
decflt_rounding	No	No	No	None	SQLF_DBTN_DECFLT_ROUNDING	913	UInt16	"decflt_rounding - Decimal floating point rounding configuration parameter" on page 148

Table 3. Configurable Database Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
dec_arithmetic	No	No	No	High	SQLF_DBTN_DEC_ARITHMETIC	10173	char(7)	"dec_arithmetic - DECIMAL arithmetic mode" on page 146
dec_to_char_fmt	Yes	Yes	Yes	Medium	SQLF_DBTN_DEC_TO_CHAR_FMT	<ul style="list-style-type: none"> • 0: V95 • 1: NEW 	Uint16	"dec_to_char_fmt - Decimal to character function configuration parameter" on page 147
dft_degree	Yes	No	Yes	High	SQLF_DBTN_DFT_DEGREE	301	Sint32	"dft_degree - Default degree" on page 150
dft_extent_sz	Yes	No	No	Medium	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32	"dft_extent_sz - Default extent size of table spaces" on page 151
dft_loadrec_ses	Yes	No	Yes	Medium	SQLF_DBTN_DFT_LOADREC_SES	42	Sint16	"dft_loadrec_ses - Default number of load recovery sessions" on page 151
dft_mttb_types	No	No	No	None	SQLF_DBTN_DFT_MTTB_TYPES	843	Uint32	"dft_mttb_types - Default maintained table types for optimization" on page 152
dft_prefetch_sz	Yes	Yes	No	Medium	SQLF_DBTN_DFT_PREFETCH_SZ	40	Sint16	"dft_prefetch_sz - Default prefetch size" on page 152
dft_queryopt	Yes	No	Yes	Medium	SQLF_DBTN_DFT_QUERYOPT	57	Sint32	"dft_queryopt - Default query optimization class" on page 153
dft_refresh_age	No	No	No	Medium	SQLF_DBTN_DFT_REFRESH_AGE	702	char(22)	"dft_refresh_age - Default refresh age" on page 154
dft_schemas_dcc	Yes	No	No	Medium	SQLF_DBTN_DFT_SCHEMAS_DCC	10115	Uint16	"dft_schemas_dcc - Default data capture on new schemas configuration parameter" on page 155
dft_table_org	Yes	No	No	Medium	SQLF_DBTN_DFT_TABLE_ORG	10126	Uint16	"dft_table_org - Default table organization" on page 157
discover_db	Yes	No	No	Medium	SQLF_DBTN_DISCOVER	308	Uint16	"discover_db - Discover database" on page 157
dlchktime	Yes	No	No	Medium	SQLF_DBTN_DLCHKTIME	9	Uint32	"dlchktime - Time interval for checking deadlock" on page 158
enable_xmlchar	Yes	No	No	None	SQLF_DBTN_ENABLE_XMLCHAR	853	Uint32	"enable_xmlchar - Enable conversion to XML configuration parameter" on page 159
encrlib	Yes	No	No	None	SQLF_DBTN_ENCRLIB	10142	char(255)	"encrlib - Encryption library" on page 159
encropts	Yes	No	No	None	SQLF_DBTN_ENCROPTS	10141	char(255)	"encropts - Encryption options" on page 160
extended_row_sz	Yes	No	No	Medium	SQLF_DBTN_LARGE_ROW_SZ	10131	Unsigned short	"extended_row_sz - Extended row size" on page 160
failarchpath	Yes	No	No	None	SQLF_DBTN_FAILARCHPATH	826	char(243)	"failarchpath - Failover log archive path" on page 161
hadr_local_host	No	No	N/A	None	SQLF_DBTN_HADR_LOCAL_HOST	811	char(256)	"hadr_local_host - HADR local host name" on page 162
hadr_local_svc	No	No	N/A	None	SQLF_DBTN_HADR_LOCAL_SVC	812	char(41)	"hadr_local_svc - HADR local service name" on page 163
hadr_peer_window	No	No	N/A	Low (see Note 3)	SQLF_DBTN_HADR_PEER_WINDOW	914	Uint32	"hadr_peer_window - HADR peer window configuration parameter" on page 163
hadr_remote_host	No	No	N/A	None	SQLF_DBTN_HADR_REMOTE_HOST	813	char(2048)	"hadr_remote_host - HADR remote host name" on page 165
hadr_remote_inst	No	No	N/A	None	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)	"hadr_remote_inst - HADR instance name of the remote server" on page 165

Table 3. Configurable Database Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
hadr_remote_svc	No	No	N/A	None	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)	"hadr_remote_svc - HADR remote service name" on page 166
hadr_replay_delay	Yes	No	N/A	None	SQLF_DBTN_HADR_REPLAY_DELAY	10119	Sint32	"hadr_replay_delay - HADR replay delay configuration parameter" on page 167
hadr_spool_limit	Yes	No	N/A	None	SQLF_DBTN_HADR_SPOOL_LIMIT	10112	Sint32	"hadr_spool_limit - HADR log spool limit configuration parameter" on page 168
hadr_ssl_label	Yes	No	No	Medium	SQLF_DBTN_HADR_SSL_LABEL	10170	char(127)	"HADR_SSL_LABEL - Label name in the key file for SSL communication between HADR primary and standby instances configuration parameter" on page 169
hadr_syncmode	No	No	N/A	None	SQLF_DBTN_HADR_SYNCMODE	817	UInt32	"hadr_syncmode - HADR synchronization mode for log writes in peer state" on page 169
hadr_target_list	Yes	No	N/A	None	SQLF_DBTN_HADR_TARGET_LIST	10114	char(2048)	"hadr_target_list - HADR target list database configuration parameter" on page 172
hadr_timeout	No	No	N/A	None	SQLF_DBTN_HADR_TIMEOUT	816	UInt32	"hadr_timeout - HADR timeout value" on page 174
indexrec ²	Yes	No	No	Medium	SQLF_DBTN_INDEXREC	30	UInt16	"indexrec - Index re-creation time" on page 65
locklist	Yes	Yes	Yes	High when it affects escalation	SQLF_DBTN_LOCK_LIST	704	UInt64	"locklist - Maximum storage for lock list" on page 176
locktimeout	No	No	Yes	Medium	SQLF_DBTN_LOCKTIMEOUT	34	Sint16	"locktimeout - Lock timeout" on page 179
log_appl_info	No	No	N/A	Low	SQLF_DBTN_LOG_APPL_INFO	10111	UInt32	"log_appl_info - Application information log record database configuration parameter" on page 180
log_ddl_stmts	Yes	No	No	Medium	SQLF_DBTN_LOG_DDL_STMTS	10110	UInt32	"log_ddl_stmts - Log Data Definition Language (DDL) statements database configuration parameter" on page 180
logarchcompr1	Yes	No	No	None	SQLF_DBTN_LOGARCHCOMPR1	10123	char(252)	"logarchcompr1 - Primary archived log file compression configuration parameter" on page 181
logarchcompr2	Yes	No	No	None	SQLF_DBTN_LOGARCHCOMPR2	10124	char(252)	"logarchcompr2 - Secondary archived log file compression configuration parameter" on page 181
logarchmeth1	Yes	No	No	None	SQLF_DBTN_LOGARCHMETH1	822	char(252)	"logarchmeth1 - Primary log archive method" on page 182
logarchmeth2	Yes	No	No	None	SQLF_DBTN_LOGARCHMETH2	823	char(252)	"logarchmeth2 - Secondary log archive method" on page 183
logarchopt1	Yes	No	No	None	SQLF_DBTN_LOGARCHOPT1	824	char(243)	"logarchopt1 - Primary log archive options" on page 185
logarchopt2	Yes	No	No	None	SQLF_DBTN_LOGARCHOPT2	825	char(243)	"logarchopt2 - Secondary log archive options" on page 185
logbufsz	No	No	Yes	High	SQLF_DBTN_LOGBUFSZ	33	UInt16	"logbufsz - Log buffer size" on page 186
logfilsiz	No	No	No	Medium	SQLF_DBTN_LOGFILSIZ	92	UInt32	"logfilsiz - Size of log files" on page 187

Table 3. Configurable Database Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
logindexbuild	Yes	No	Yes	None	SQLF_DBTN_LOGINDEXBUILD	818	UInt32	"logindexbuild - Log index pages created" on page 188
logprimary	No	No	No	Medium	SQLF_DBTN_LOGPRIMARY	16	UInt16	"logprimary - Number of primary log files" on page 189
logsecond	Yes	No	No	Medium	SQLF_DBTN_LOGSECOND	17	UInt16	"logsecond - Number of secondary log files" on page 190
max_log	Yes	Yes	Yes	Medium	SQLF_DBTN_MAX_LOG	807	UInt16	"max_log - Maximum log per transaction" on page 192
maxappls	Yes	Yes	Yes	Medium	SQLF_DBTN_MAXAPPLS	6	UInt16	"maxappls - Maximum number of active applications" on page 193
maxfilop	Yes	No	Yes	Medium	SQLF_DBTN_MAXFILOP	3	UInt16	"maxfilop - Maximum database files open per database" on page 194
maxlocks	Yes	Yes	Yes	High when it affects escalation	SQLF_DBTN_MAXLOCKS	15	UInt16	"maxlocks - Maximum percent of lock list before escalation" on page 195
min_dec_div_3 (deprecated)	No	No	No	High	SQLF_DBTN_MIN_DEC_DIV_3	605	Sint32	"min_dec_div_3 - Decimal division scale to 3" on page 196
mincommit (deprecated)	Yes	No	Yes	High	SQLF_DBTN_MINCOMMIT	32	UInt16	"mincommit - Number of commits to group" on page 198
mirrorlogpath	No	No	No	Low	SQLF_DBTN_MIRRORLOGPATH	806	char(242)	"mirrorlogpath - Mirror log path" on page 199
mon_act_metrics	Yes	No	Yes	Medium	SQLF_DBTN_MON_ACT_METRICS	931	UInt16	"mon_act_metrics - Monitoring activity metrics configuration parameter" on page 201
mon_deadlock	Yes	No	Yes	Medium	SQLF_DBTN_MON_DEADLOCK	934	UInt16	"mon_deadlock - Monitoring deadlock configuration parameter" on page 202
mon_locktimeout	Yes	No	Yes	Medium	SQLF_DBTN_MON_LOCKTIMEOUT	933	UInt16	"mon_locktimeout - Monitoring lock timeout configuration parameter" on page 203
mon_lockwait	Yes	No	Yes	Medium	SQLF_DBTN_MON_LOCKWAIT	935	UInt16	"mon_lockwait - Monitoring lock wait configuration parameter" on page 204
mon_lw_thresh	Yes	No	Yes	Medium	SQLF_DBTN_MON_LW_THRESH	936	UInt32	"mon_lw_thresh - Monitoring lock wait threshold configuration parameter" on page 205
mon_lck_msg_lvl	Yes	No	Yes	None	SQLF_DBTN_MON_LCK_MSG_LVL	951	UInt16	"mon_lck_msg_lvl - Monitoring lock event notification messages configuration parameter" on page 205
mon_obj_metrics	Yes	No	Yes	Medium	SQLF_DBTN_MON_OBJ_METRICS	937	UInt16	"mon_obj_metrics - Monitoring object metrics configuration parameter" on page 205
mon_pkglist_sz	Yes	No	Yes	Low	SQLF_DBTN_MON_PKGLIST_SZ	950	UInt32	"mon_pkglist_sz - Monitoring package list size configuration parameter" on page 208
mon_req_metrics	Yes	No	Yes	Medium	SQLF_DBTN_MON_REQ_METRICS	930	UInt16	"mon_req_metrics - Monitoring request metrics configuration parameter" on page 208

Table 3. Configurable Database Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
mon_rtn_data	Yes	No	No	Medium	SQLF_DBTN_MON_RTN_DATA	10130	UInt16	"mon_rtn_data - Monitoring routine capture" on page 209
mon_rtn_execlist	Yes	No	No	Medium	SQLF_DBTN_MON_RTN_EXECLIST	10128	UInt16	"mon_rtn_execlist - Monitoring routine executable list" on page 210
mon_uow_data	Yes	No	Yes	Medium	SQLF_DBTN_MON_UOW_DATA	932	UInt16	"mon_uow_data - Monitoring unit of work events configuration parameter" on page 211
mon_uow_execlist	Yes	No	Yes	Medium	SQLF_DBTN_MON_UOW_EXECLIST	957	UInt16	"mon_uow_execlist - Monitoring unit of work events with executable list configuration parameter" on page 212
mon_uow_pkglist	Yes	No	Yes	Medium	SQLF_DBTN_MON_UOW_PKGLIST	956	UInt16	"mon_uow_pkglist - Monitoring unit of work events with package list configuration parameter" on page 212
nchar_mapping	Yes	No	No	Low	SQLF_DBTN_NCHAR_MAPPING	10133	UInt16	"nchar_mapping - National character mapping" on page 213
newlogpath	No	No	No	Low	SQLF_DBTN_NEWLOGPATH	20	char(242)	"newlogpath - Change the database log path" on page 214
num_db_backups	Yes	No	No	None	SQLF_DBTN_NUM_DB_BACKUPS	601	UInt16	"num_db_backups - Number of database backups" on page 215
num_freqvalues	Yes	No	No	Low	SQLF_DBTN_NUM_FREQVALUES	36	UInt16	"num_freqvalues - Number of frequent values retained" on page 216
num_iocleaners	No	Yes	Yes	High	SQLF_DBTN_NUM_IOCLEANERS	37	UInt16	"num_iocleaners - Number of asynchronous page cleaners" on page 217
num_ioservers	No	Yes	Yes	High	SQLF_DBTN_NUM_IOSERVERS	39	UInt16	"num_ioservers - Number of I/O servers" on page 218
num_log_span	Yes	Yes	Yes		SQLF_DBTN_NUM_LOG_SPAN	808	UInt16	"num_log_span - Number log span" on page 219
num_quantiles	Yes	No	Yes	Low	SQLF_DBTN_NUM_QUANTILES	48	UInt16	"num_quantiles - Number of quantiles for columns" on page 220
numarchretry	Yes	No	Yes	None	SQLF_DBTN_NUMARCHRETRY	827	UInt16	"numarchretry - Number of retries on error" on page 221
overflowlogpath	Yes	No	No	Medium	SQLF_DBTN_OVERFLOWLOGPATH	805	char(242)	"overflowlogpath - Overflow log path" on page 223
page_age_trgt_gcr	No	No	No	High	SQLF_DBTN_PAGE_AGE_TARGET_GCR	10136	Unsigned short	"page_age_trgt_gcr - Page age target group crash recovery configuration parameter" on page 224
page_age_trgt_mcr	No	No	No	High	SQLF_DBTN_PAGE_AGE_TARGET_MCR	10137	Unsigned short	"page_age_trgt_mcr - Page age target member crash recovery configuration parameter" on page 225
pckcachesz	Yes	Yes	Yes	High	SQLF_DBTN_PCKCACHE_SZ	505	UInt32	"pckcachesz - Package cache size" on page 225
pl_stack_trace	Yes	No	No	Low	SQLF_DBTN_PL_STACK_TRACE	10168	UInt16	"pl_stack_trace" on page 227
rec_his_retentn	No	No	No	None	SQLF_DBTN_REC_HIS_RETENTN	43	Sint16	"rec_his_retentn - Recovery history retention period" on page 228
section_actuals	Yes	No	No	High	SQLF_DBTN_SECTION_ACTUALS	952	UInt64	"section_actuals - Section actuals configuration parameter" on page 230

Table 3. Configurable Database Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
self_tuning_mem	Yes	No	Yes	High	SQLF_DBTN_SELF_TUNING_MEM	848	UInt16	"self_tuning_mem - Self-tuning memory" on page 231
seqdetect	Yes	No	No	High	SQLF_DBTN_SEQDETECT	41	UInt16	"seqdetect - Sequential detection and readahead flag" on page 232
sheapthres_shr	Yes	Yes	Yes	High	SQLF_DBTN_SHEAPTHRES_SHR	802	UInt32	"sheapthres_shr - Sort heap threshold for shared sorts" on page 233
smtp_server	Yes	No	Yes	None	SQLF_DBTN_SMTP_SERVER	926	char [] (String)	"smtp_server - SMTP server" on page 237
softmax ₁	No	No	No	Medium	SQLF_DBTN_SOFTMAX	5	UInt16	"softmax - Recovery range and soft checkpoint interval" on page 237
sortheap	Yes	Yes	Yes	High	SQLF_DBTN_SORT_HEAP	52	UInt32	"sortheap - Sort heap size" on page 239
sql_ccflags	Yes	No	Yes	None	SQLF_DBTN_SQL_CCFLAGS	927	char(1023)	"sql_ccflags - Conditional compilation flags" on page 243
stat_heap_sz	Yes	Yes	Yes	Low	SQLF_DBTN_STAT_HEAP_SZ	45	UInt32	"stat_heap_sz - Statistics heap size" on page 244
stmt_conc	Yes	No	Yes	Medium	SQLF_DBTN_STMT_CONC	919	UInt32	"stmt_conc - Statement concentrator configuration parameter" on page 244
stmtheap	Yes	Yes	Yes	Medium	SQLF_DBTN_STMT_HEAP	821	UInt32	"stmtheap - Statement heap size" on page 246
string_units	Yes	No	No	Low	SQLF_DBTN_STRING_UNITS	10132	UInt16	"string_units - Default string units" on page 247
sysptime_period_adj	Yes	No	No	None	SQLF_DBTN_SYSTIME_PERIOD_ADJ	955	UInt16	"sysptime_period_adj - Adjust temporal SYSTEM_TIME period database configuration parameter" on page 248
trackmod	No	No	No	Low	SQLF_DBTN_TRACKMOD	703	UInt16	"trackmod - Track modified pages enable" on page 250
tsm_mgmtclass	Yes	No	Yes	None	SQLF_DBTN_TSM_MGMTCLASS	307	char(30)	"tsm_mgmtclass - Tivoli Storage Manager management class" on page 250
tsm_nodename	Yes	No	Yes	None	SQLF_DBTN_TSM_NODENAME	306	char(64)	"tsm_nodename - Tivoli Storage Manager node name" on page 250
tsm_owner	Yes	No	Yes	None	SQLF_DBTN_TSM_OWNER	305	char(64)	"tsm_owner - Tivoli Storage Manager owner name" on page 251
tsm_password	Yes	No	Yes	None	SQLF_DBTN_TSM_PASSWORD	501	char(64)	"tsm_password - Tivoli Storage Manager password" on page 251
util_heap_sz	Yes	Yes	Yes	Low	SQLF_DBTN_UTIL_HEAP_SZ	55	UInt32	"util_heap_sz - Utility heap size" on page 252
vendoropt	Yes	No	No	None	SQLF_DBTN_VENDOROPT	829	char(242)	"vendoropt - Vendor options" on page 254<
wlm_collect_int	Yes	No	Yes	Low	SQLF_DBTN_WLM_COLLECT_INT	907	Sint32	"wlm_collect_int - Workload management collection interval configuration parameter" on page 255
wlm_cpu_limit	Yes	No	Yes	Medium	SQLF_DBTN_WLM_CPU_LIMIT	10166	UInt16	"wlm_cpu_limit - WLM CPU limit" on page 256
wlm_cpu_shares	Yes	No	Yes	Medium	SQLF_DBTN_WLM_CPU_SHARES	10164	UInt16	"wlm_cpu_shares - WLM CPU shares" on page 256
wlm_cpu_share_mode	Yes	No	Yes	Medium	SQLF_DBTN_WLM_CPU_SHARE_MODE	10165	UInt16	"wlm_cpu_share_mode - WLM CPU share mode" on page 256

Table 3. Configurable Database Configuration Parameters (continued)

Parameter	Cfg. Online	Auto.	Member Cfg.	Perf. Impact	Token	Token Value	Data Type	Additional Information
<p>¹ Important: The softmax database configuration parameter is deprecated is deprecated in Version 10.5 and might be removed in a future release. For more information, see Some database configuration parameters are deprecated in <i>What's New for Db2 Version 10.5</i>. Note: Refer to the header files <code>sqlenv.h</code> and <code>sqlutil.h</code> for the valid values and for the definitions that are used by the configuration parameters.</p> <p>1. The bits of <code>SQLF_DBTN_AUTONOMIC_SWITCHES</code> indicate the default settings for a number of auto-maintenance configuration parameters. The individual bits making up this composite parameter are: Default => Bit 1 on (xxxx xxxx xxxx xxx1): <code>auto_maint</code> Bit 2 off (xxxx xxxx xxxx xx0x): <code>auto_db_backup</code> Bit 3 on (xxxx xxxx xxxx x1xx): <code>auto_tbl_maint</code> Bit 4 on (xxxx xxxx xxxx lxxx): <code>auto_runstats</code> Bit 5 off (xxxx xxxx xxx0 xxxx): <code>auto_stats_prof</code> Bit 6 off (xxxx xxxx xx0x xxxx): <code>auto_prof_upd</code> Bit 7 off (xxxx xxxx x0xx xxxx): <code>auto_reorg</code> Bit 8 off (xxxx xxxx 0xxx xxxx): <code>auto_storage</code> Bit 9 off (xxxx xxx0 xxxx xxxx): <code>auto_stmt_stats</code> 0 0 0 0</p> <p>Maximum => Bit 1 on (xxxx xxxx xxxx xxx1): <code>auto_maint</code> Bit 2 off (xxxx xxxx xxxx xx1x): <code>auto_db_backup</code> Bit 3 on (xxxx xxxx xxxx x1xx): <code>auto_tbl_maint</code> Bit 4 on (xxxx xxxx xxxx lxxx): <code>auto_runstats</code> Bit 5 off (xxxx xxxx xxx1 xxxx): <code>auto_stats_prof</code> Bit 6 off (xxxx xxxx xx1x xxxx): <code>auto_prof_upd</code> Bit 7 off (xxxx xxxx x1xx xxxx): <code>auto_reorg</code> Bit 8 off (xxxx xxxx lxxx xxxx): <code>auto_storage</code> Bit 9 off (xxxx xxx1 xxxx xxxx): <code>auto_stmt_stats</code> 0 1 F F</p> <p>Note: The auto_stats_prof and auto_prof_upd parameters are discontinued in Version 10.5. For more information, see Some database configuration parameters are deprecated or discontinued in <i>What's New for Db2 Version 10.5</i>.</p> <p>2. Valid values <code>SQLF_INX_REC_SYSTEM</code> (0) <code>SQLF_INX_REC_REFERENCE</code> (1) <code>SQLF_INX_REC_RESTART</code> (2) <code>SQLF_INX_REC_RESTART_NO_REDO</code> (3) <code>SQLF_INX_REC_ACCESS_NO_REDO</code> (4)</p> <p>3. If you set the hadr_peer_window parameter to a nonzero time value, the primary hangs on transactions when it is in disconnected peer state. This hang occurs because the primary database is waiting for confirmation from the standby database even though it is not connected to the standby database.</p>								

Table 4. Informational Database Configuration Parameters

Parameter	Token	Token Value	Data Type	Additional Information
backup_pending	<code>SQLF_DBTN_BACKUP_PENDING</code>	112	UInt16	"backup_pending - Backup pending indicator" on page 126
codepage	<code>SQLF_DBTN_CODEPAGE</code>	101	UInt16	"codepage - Code page for the database" on page 134
codeset	<code>SQLF_DBTN_CODESET</code>	120	char(9) ¹	"codeset - Codeset for the database" on page 134
collate_info	<code>SQLF_DBTN_COLLATE_INFO</code>	44	char(260)	"collate_info - Collating information" on page 134
country/region	<code>SQLF_DBTN_COUNTRY</code>	100	UInt16	"country/region - Database territory code" on page 136
database_consistent	<code>SQLF_DBTN_CONSISTENT</code>	111	UInt16	"database_consistent - Database is consistent" on page 137
database_level	<code>SQLF_DBTN_DATABASE_LEVEL</code>	124	UInt16	"database_level - Database release level" on page 138
date_compat	<code>SQLF_DBTN_DATE_COMPAT</code>	918	UInt16	"date_compat - Date compatibility database configuration parameter" on page 47
hadr_db_role	<code>SQLF_DBTN_HADR_DB_ROLE</code>	810	UInt32	"hadr_db_role - HADR database role" on page 162

Table 4. Informational Database Configuration Parameters (continued)

Parameter	Token	Token Value	Data Type	Additional Information
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	UInt16	"log_retain_status - Log retain status indicator" on page 180
loghead	SQLF_DBTN_LOGHEAD	105	char(12)	"loghead - First active log file" on page 188
logpath	SQLF_DBTN_LOGPATH	103	char(242)	"logpath - Location of log files" on page 189
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	UInt16	"multipage_alloc - Multipage file allocation enabled" on page 213
numsegs (deprecated)	SQLF_DBTN_NUMSEGS	122	UInt16	"numsegs - Default number of SMS containers" on page 222
pagesize	SQLF_DBTN_PAGESIZE	846	UInt32	"pagesize - Database default page size" on page 225
release	SQLF_DBTN_RELEASE	102	UInt16	"release - Configuration file release level" on page 89
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	UInt16	"restore_pending - Restore pending" on page 229
restrict_access	SQLF_DBTN_RESTRICT_ACCESS	852	Sint32	"restrict_access - Database has restricted access configuration parameter" on page 229
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	UInt16	"rollfwd_pending - Roll forward pending indicator" on page 230
suspend_io	SQLF_DBTN_SUSPEND_IO	953	UInt16	"suspend_io - Database I/O operations state configuration parameter" on page 248
territory	SQLF_DBTN_TERRITORY	121	char(5) ²	"territory - Database territory" on page 249
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	UInt16	"user_exit_status - User exit status indicator" on page 252
<p>Note:</p> <ol style="list-style-type: none"> 1. char(17) on AIX®, HP-UX, Linux and Solaris operating systems. 2. char(33) on AIX, HP-UX, Linux and Solaris operating systems. 				

Db2 Administration Server (DAS) Configuration Parameter Summary

Important: The Db2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in Db2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see "Db2 administration server (DAS) has been deprecated" at .

Table 5. DAS Configuration Parameters

Parameter	Parameter Type	Additional Information
authentication	Configurable	"authentication - Authentication type DAS" on page 257
contact_host	Configurable Online	"contact_host - Location of contact list" on page 258
das_codepage	Configurable Online	"das_codepage - DAS code page" on page 258
das_territory	Configurable Online	"das_territory - DAS territory" on page 259
dasadm_group	Configurable	sy
db2system	Configurable Online	"db2system - Name of the Db2 server system" on page 260
discover	Configurable Online	"discover - DAS discovery mode" on page 260
exec_exp_task	Configurable	"exec_exp_task - Execute expired tasks" on page 261
jdk_64_path	Configurable Online	"jdk_64_path - 64-Bit Software Developer's Kit for Java installation path DAS" on page 261
jdk_path	Configurable Online	"jdk_path - Software Developer's Kit for Java installation path DAS" on page 262
sched_enable	Configurable	"sched_enable - Scheduler mode" on page 262

Table 5. DAS Configuration Parameters (continued)

Parameter	Parameter Type	Additional Information
sched_userid	Informational	"sched_userid - Scheduler user ID" on page 263
smtp_server	Configurable Online	"smtp_server - SMTP server" on page 263
toolscat_db	Configurable	"toolscat_db - Tools catalog database" on page 263
toolscat_inst	Configurable	"toolscat_inst - Tools catalog database instance" on page 264
toolscat_schema	Configurable	"toolscat_schema - Tools catalog database schema" on page 264

Ingest Utility Configuration Parameter Summary

Table 6. Ingest Utility Configuration Parameters

Parameter	Parameter Type	Additional Information
commit_count	Configurable	"commit_count - Commit count configuration parameter" on page 265
commit_period	Configurable	"commit_period - Commit period configuration parameter" on page 266
num_flushers_per_partition	Configurable	"num_flushers_per_partition - Number of flushers per database partition configuration parameter" on page 266
num_formatters	Configurable	"num_formatters - Number of formatters configuration parameter" on page 267
pipe_timeout	Configurable	"pipe_timeout - Pipe timeout configuration parameter" on page 267
retry_count	Configurable	"retry_count - Retry count configuration parameter" on page 267
retry_period	Configurable	"retry_period - Retry period configuration parameter" on page 268
shm_max_size	Configurable	"shm_max_size - Maximum size of shared memory configuration parameter" on page 268

Configuration parameter section headings

Each of the configuration parameter descriptions contains some or all of the following section headings, as applicable. In some cases, they are mutually exclusive, for example, valid values are not needed if the [range] is specified. In most cases, these headings are self-explanatory.

Table 7. Description of the configuration parameter section headings

Section heading	Description and possible values
Configuration type	Possible values are: <ul style="list-style-type: none"> • Database manager • Database • Db2 Administration Server
Applies to	If applicable, lists the data server types that the configuration parameter applies to. Possible values are: <ul style="list-style-type: none"> • Client • Database server with local and remote clients • Database server with local clients • Db2 Administration Server • OLAP functions • Partitioned database server with local and remote clients • Partitioned database server with local and remote clients when federation is enabled. • Satellite database server with local clients
Parameter type	Possible values are: <ul style="list-style-type: none"> • Configurable (the database manager must be restarted to have the changes take effect) • Configurable online (can be dynamically updated online without having to restart the database manager) • Informational (values are for your information only and cannot be updated) • Configurable by member in a Db2 pureScale environment

Table 7. Description of the configuration parameter section headings (continued)

Section heading	Description and possible values
Default [range]	If applicable, lists the default value and the possible ranges, including NULL values or automatic settings. If the range differs by platform, then the values are listed by platform or platform type, for example, 32-bit or 64-bit platforms. In most cases the default value is not listed as part of the range.
Unit of measure	If applicable, lists the unit of measure. Possible values are: <ul style="list-style-type: none"> • Bytes • Counter • Megabytes per second • Milliseconds • Minutes • Pages (4 KB) • Percentage • Seconds
Valid values	If applicable, lists the valid value. This heading is mutually exclusive with the default [range] heading.
Examples	If applicable, lists examples.
Propagation class	If applicable, possible values are: <ul style="list-style-type: none"> • Immediate • Statement boundary • Transaction boundary • Connection
When allocated	If applicable, indicates when the configuration parameter is allocated by the database manager.
When freed	If applicable, indicates when the configuration parameter is freed by the database manager.
Restrictions	If applicable, lists any restrictions that apply to the configuration parameter.
Limitations	If applicable, lists any limitations that apply to the configuration parameter.
Recommendations	If applicable, lists any recommendations that apply to the configuration parameter.
Usage notes	If applicable, lists any usage notes that apply to the configuration parameter.

Configuration parameters that affect the number of agents

There are a number of database manager configuration parameters related to database agents and how they are managed.

The following database manager configuration parameters determine how many database agents are created and how they are managed:

- **Agent Pool Size (num_poolagents):** The total number of idle agents to pool that are kept available in the system. The default value for this parameter is 100, AUTOMATIC.
- **Initial Number of Agents in Pool (num_initagents):** When the database manager is started, a pool of worker agents is created based on this value. This speeds up performance for initial queries. All worker agents begin as idle agents.
- **Maximum Number of Connections (max_connections):** Specifies the maximum number of connections allowed to the database manager system on each database partition.
- **Maximum Number of Coordinating Agents (max_coordagents):** For partitioned database environments and environments with intrapartition parallelism enabled, when **Connection concentrator** is enabled, this value limits the number of coordinating agents.

Configuration parameters that affect query optimization

Several configuration parameters affect the access plan chosen by the SQL or XQuery compiler. Many of these are appropriate to a single-partition database environment and some are only appropriate to a partitioned database environment.

Assuming a homogeneous partitioned database environment, where the hardware is the same, the values used for each parameter should be the same on all database partitions.

Note: When you change a configuration parameter dynamically, the optimizer might not read the changed parameter values immediately because of older access plans in the package cache. To reset the package cache, execute the **FLUSH PACKAGE CACHE** command.

In a federated system, if the majority of your queries access nicknames, evaluate the types of queries that you send before you change your environment. For example, in a federated database, the buffer pool does not cache pages from data sources, which are the DBMSs and data within the federated system. For this reason, increasing the size of the buffer does not guarantee that the optimizer considers additional access-plan alternatives when it chooses an access plan for queries that contain nicknames. However, the optimizer might decide that local materialization of data source tables is the least-cost route or a necessary step for a sort operation. In that case, increasing the resources available might improve performance.

The following configuration parameters or factors affect the access plan chosen by the SQL or XQuery compiler:

- The size of the buffer pools that you specified when you created or altered them.

When the optimizer chooses the access plan, it considers the I/O cost of fetching pages from disk to the buffer pool and estimates the number of I/Os required to satisfy a query. The estimate includes a prediction of buffer pool usage, because additional physical I/Os are not required to read rows in a page that is already in the buffer pool.

The optimizer considers the value of the `npages` column in the `SYSCAT.BUFFERPOOLS` system catalog tables and, in partitioned database environments, the `SYSCAT.BUFFERPOOLDBPARTITIONS` system catalog tables.

The I/O costs of reading the tables can have an impact on how two tables are joined and if an unclustered index is used to read the data

- Default Degree (**dft_degree**)

The **dft_degree** configuration parameter specifies parallelism by providing a default value for the **CURRENT DEGREE** special register and the `DEGREE` bind option. A value of one (1) means no intrapartition parallelism. A value of minus one (-1) means the optimizer determines the degree of intrapartition parallelism based on the number of processors and the type of query.

Note: Intra-parallel processing does not occur unless you enable it by setting the **intra_parallel** database manager configuration parameter.

- Default Query Optimization Class (**dft_queryopt**)

Although you can specify a query optimization class when you compile SQL or XQuery queries, you can also set a default query optimization class.

- Average Number of Active Applications (**avg_apps**)

The optimizer uses the **avg_appls** parameter to help estimate how much of the buffer pool might be available at run-time for the access plan chosen. Higher values for this parameter can influence the optimizer to choose access plans that are more conservative in buffer pool usage. If you specify a value of one (1), the optimizer considers that the entire buffer pool is available to the application.

- Sort Heap Size (**sortheap**)

If the rows to be sorted occupy more than the space available in the sort heap, several sort passes are performed, where each pass sorts a subset of the entire set of rows. Each sort pass is stored in a system temporary table in the buffer pool, which might be written to disk. When all the sort passes are complete, these sorted subsets are merged into a single sorted set of rows. A sort that does not require a system temporary table to store the list of data always results in better performance and is used if possible.

When choosing an access plan, the optimizer estimates the cost of the sort operations, including evaluating whether a sort can be read in a single, sequential access, by estimating the amount of data to be sorted and looking at the **sortheap** parameter to determine if there is enough space to read a sort in a single, sequential access.

- Maximum Storage for Lock List (**locklist**) and Maximum Percent of Lock List Before Escalation (**maxlocks**)

When the isolation level is repeatable read (RR), the optimizer considers the values of the **locklist** and **maxlocks** parameters to determine whether row level locks might be escalated to a table level lock. If the optimizer estimates that lock escalation might occur for a table access, then it chooses a table level lock for the access plan, instead of incurring the overhead of lock escalation during the query execution.

- CPU Speed (**cpuspeed**)

The optimizer uses the CPU speed to estimate the cost of performing certain operations. CPU cost estimates and various I/O cost estimates help select the best access plan for a query.

The CPU speed of a machine can have a significant influence on the access plan chosen. This configuration parameter is automatically set to an appropriate value when the database is installed or upgraded. Do not adjust this parameter unless you are modelling a production environment on a test system or assessing the impact of a hardware change. Using this parameter to model a different hardware environment allows you to find out the access plans that might be chosen for that environment. To have the database manager recompute the value of this automatic configuration parameter, set it to minus one (-1).

- Statement Heap Size (**stmheap**)

Although the size of the statement heap does not influence the optimizer in choosing different access paths, it can affect the amount of optimization performed for complex SQL or XQuery statements.

If the **stmheap** parameter is not set to a sufficient value, you might receive a warning indicating that there is not enough memory available to process the statement. For example, SQLCODE +437 (SQLSTATE 01602) might indicate that the amount of optimization that has been used to compile a statement is less than the amount that you requested.

- Communications Bandwidth (**comm_bandwidth**)

Communications bandwidth is used by the optimizer to determine access paths. The optimizer uses the value in this parameter to estimate the cost of performing certain operations between the database partition servers in a partitioned database environment.

- Application Heap Size (**applheapsz**)
Large schemas require sufficient space in the application heap.

Recompiling a query after configuration changes

To observe the effect of configuration changes that affect query optimization, it might be necessary to cause the query optimizer to recompile the statements that are cached.

Procedure

You can cause the query optimizer to recompile a statement by performing any of the following actions:

- Invalidating the cached dynamic statements for specific tables using the **RUNSTATS** command:

```
RUNSTATS ON TABLE <tableschema>.<tablename>  
WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL
```

Note: This will refresh the table statistics and subsequent query compilations will use the new statistics as well as the new configuration settings.

- Removing all cached dynamic SQL statements currently in the package cache:
FLUSH PACKAGE CACHE DYNAMIC

Related reference:

RUNSTATS command

“Configuration parameters summary” on page 5

The following tables list the parameters in the database manager and database configuration files for database servers. When you change the database manager and database configuration parameters, consider the detailed information for each parameter. Specific operating environment information (including defaults) is part of each parameter description.

Restrictions and behavior when configuring **max_coordagents** and **max_connections**

The Version 9.5 default for the **max_coordagents** and **max_connections** parameters will be **AUTOMATIC**, with **max_coordagents** set to 200 and **max_connections** set to -1 (that is, set to the value of **max_coordagents**). These settings set Concentrator to **OFF**.

While configuring **max_coordagents** or **max_connections** online, there will be some restrictions and behavior to be aware of:

- If the value of **max_coordagents** is increased, the setting takes effect immediately and new requests will be allowed to create new coordinating agents. If the value is decreased, the number of coordinating agents will not be reduced immediately. Rather, the number of coordinating agents will no longer increase, and existing coordinating agents might terminate after finishing their current set of work, in order to reduce the overall number of coordinating agents. New requests for work that require a coordinating agent are not serviced until the total number of coordinating agents is less than the new value and a coordinating agent becomes free.
- If the value for **max_connections** is increased, the setting takes effect immediately and new connections previously blocked because of this parameter will be allowed. If the value is decreased, the database manager will not actively

terminate existing connections; instead, new connections will not be allowed until enough of the existing connections are terminated so that the value is less than the new maximum.

- If **max_connections** is set to -1 (default), then the maximum number of connections allowed is the same as **max_coordagents**, and when **max_coordagents** is updated offline or online; the maximum number of connections allowed will be updated as well.

While changing the value of **max_coordagents** or **max_connections** online, you cannot change it such that connection Concentrator will be turned either ON, if it's off, or OFF, if it's ON. For example, if at **START DBM** time **max_coordagents** is less than **max_connections** (Concentrator is ON), then all updates done online to these two parameters must maintain the relationship **max_coordagents** < **max_connections**. Similarly, if at **START DBM** time, **max_coordagents** is greater than or equal to **max_connections** (Concentrator is OFF), then all updates done online must maintain this relationship.

When you perform this type of update online, the database manager does not fail the operation, instead it defers the update. The warning SQL1362W message is returned, similar to any case when updating the database manager configuration parameters where **IMMEDIATE** is specified, but is not possible.

When setting **max_coordagents** or **max_connections** to **AUTOMATIC**, the following behavior can be expected:

- Both of these parameters can be configured with a starting value and an **AUTOMATIC** setting. For example, the following command associates a value of 200 and **AUTOMATIC** to the **max_coordagents** parameter:

```
UPDATE DBM CONFIG USING max_coordagents 200 AUTOMATIC
```

These parameters will always have a value associated with them, either the value set as default, or some value that you specified. If only **AUTOMATIC** is specified when updating either parameter, that is, no value is specified, and the parameter previously had a value associated with it, that value would remain. Only the **AUTOMATIC** setting would be affected.

Note: When Concentrator is ON, the values assigned to these two configuration parameters are important even when the parameters are set to **AUTOMATIC**.

- If both parameters are set to **AUTOMATIC**, the database manager allows the number of connections and coordinating agents to increase as needed to suit the workload. However, the following caveats apply:
 1. When Concentrator is OFF, the database manager maintains a one-to-one ratio: for every connection there will be only *one* coordinating agent.
 2. When Concentrator is ON, the database manager tries to maintain the ratio of coordinating agents to connections set by the values in the parameters.

Note:

- The approach used to maintain the ratio is designed to be unintrusive and does not guarantee the ratio will be maintained perfectly. New connections are always allowed in this scenario, though they might have to wait for an available coordinating agent. New coordinating agents will be created as needed to maintain the ratio. As connections are terminated, the database manager might also terminate coordinating agents to maintain the ratio

- The database manager will not reduce the ratio that you set. The initial values of **max_coordagents** and **max_connections** that you set are considered a lower bound.
- The current and delayed values of both these parameters can be displayed through various means, such as CLP or APIs. The values displayed will always be the values set by the user. For example, if the following command were issued, and then 30 concurrent connections performing work on the instance were started, the displayed values for **max_connections** and **max_coordagents** will still be 20, AUTOMATIC:

```
UPDATE DBM CFG USING max_connections 20 AUTOMATIC,
max_coordagents 20 AUTOMATIC
```

To determine the real number of connections and coordinating agents currently running monitor elements, you can also use the Health Monitor.

- If **max_connections** is set to AUTOMATIC with a value greater than **max_coordagents** (so that Concentrator is ON), and **max_coordagents** is not set to AUTOMATIC, then the database manager allows an unlimited number of connections that will use only a limited number of coordinating agents.

Note: Connections might have to wait for available coordinating agents.

The use of the AUTOMATIC option for the **max_coordagents** and **max_connections** configuration parameters is only valid in the following two scenarios:

1. Both parameters are set to AUTOMATIC
2. Concentrator is enabled with **max_connections** set to AUTOMATIC, while **max_coordagents** is not.

All other configurations using AUTOMATIC for these parameters will be blocked and will return SQL6112N, with a reason code that explains the valid settings of AUTOMATIC for these two parameters.

Failsafe option to update the database manager configuration file

The information within the configuration file is responsible for controlling access to essential features of your instance, including the permission to update the configuration file itself. Do not update any database manager configuration parameters without fully understanding its effect on your instance. Parameters that control access to the instance and require update with caution includes the following parameters:

- **AUTHENTICATION:** Setting this parameter to a value that is not supported by the operating system, prevents Db2 from recognizing or authenticating users. As Db2 prevents user access, all connections are ignored, and any checks for SYSADM/SYSCTRL/SYSMAINT fails (plus many other problems). Without a connection to Db2 the database manager configuration file is inaccessible and hence a proper value of **AUTHENTICATION** cannot be restored.
- **SYSADM_GROUP:** Setting this parameter to a non-existing group ensures that Db2 considers all users to be non-SYSADM, preventing usage of all commands that requires SYSADM. Without SYSADM, the database manager configuration file cannot be updated, and hence a proper value of **SYSADM_GROUP** cannot be restored.

When these features are updated improperly, your access to your own instance is impaired.

To regain your access, you require a highly privileged local operating system security user to override the database security check of Db2 to correct the database manager configuration file. For existing operating systems, this highly privileged user are the following users:

- Linux/Unix: The instance owner
- Windows: Someone that is classified as an "Administrator"

Attention: The security bypass is restricted to a *local* update of the database manager configuration file. You cannot use a fail-safe user remotely or for any other Db2 database command.

Database manager configuration parameters

You can set database manager configuration parameters to control server process, how clients connect to the database, and memory sharing and allocation on the database server. This type of parameter affects Db2 at the instance level.

agent_stack_sz - Agent stack size

You can use this parameter to determine the amount of memory that is allocated by Db2 for each agent thread stack.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Linux (32-bit)

256 [16 - 1024]

Linux (64-bit) and UNIX

1024 [256 - 32768]

Windows

16 [8 - 1000]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

On Linux and UNIX operating systems, stack space (process virtual memory) is allocated as needed or reused in the main Db2 server process when a thread is created. Stack memory is used or committed as necessary.

On Windows operating systems, **agent_stack_sz** represents the initial committed stack memory when a thread is created. Additional stack memory is used or committed as necessary.

When freed

On Linux and UNIX operating systems, stack space (process virtual memory) is retained for reuse when threads terminate and are freed when the Db2 server shuts down.

On Windows operating systems, stack space and memory are freed when a thread terminates.

On Linux and UNIX operating systems, **agent_stack_sz** is rounded up to the next largest power-of-2 based value. The default settings should be sufficient for most workloads.

On Windows operating systems, the **agent_stack_sz** configuration parameter is used to set the initial committed stack size for each agent. Regardless of the setting, each agent stack can grow to the minimum reserved stack size of 256 KB on 32-bit versions of Windows and 2 MB on 64-bit versions of Windows. If you exceed the minimum reserved stack size, the agent stack might run out of space and return an error.

Windows operating systems use the concepts of a "reserved" stack, which is the maximum to which the stack can grow, and the "committed" stack, which is the amount of memory committed to the stack when it is created. In addition, a guard page is added to the specified committed stack size in order to determine the minimum reserved stack space required. For example, with **agent_stack_sz** (committed stack) set to 16, one (1) guard page is added. This means that the reserved stack size must be at least 17 pages. The maximum, or reserved, agent stack size can be increased by setting the **agent_stack_sz** configuration parameter, which determines the size of the committed stack, to a value that results in a minimum reserved stack size larger than the default reserve stack size of 64 pages. Note that Windows operating systems use multiples of 1 MB for setting reserved stack sizes greater than 256 KB. For example, on 32-bit Windows operating systems, setting the **agent_stack_sz** configuration parameter to a value within the range of 64 - 255 4-KB pages results in a maximum stack usage of 1 MB because $(64 * 4 \text{ KB}) + 4 \text{ KB guard page} = 260 \text{ KB}$. This exceeds the 256 KB threshold so the value is rounded up to 1 MB for the reserved stack area. If you set the **agent_stack_sz** configuration parameter to 255, then the size of the reserved stack is $(255 * 4 \text{ KB}) + 4 \text{ KB guard page} = 1024 \text{ KB}$, making 255 the maximum setting before exceeding the 1 MB threshold.

You can change the default reserve stack size by using the **db2hdr** utility to change the header information for the `db2syscs.exe` file. The advantage of changing the default reserved stack size using the **db2hdr** utility is that it provides a finer granularity, therefore allowing you to set the stack size to be a minimum required stack size. This conserves virtual address space on 32-bit Windows. However, you must restart Db2 for a change to the `db2syscs.exe` process to take effect, and this method must be repeated with any Fix Pack upgrade.

Recommendations:

If you are working with large or complex XML data in a 32-bit Windows operating system, you should update the value of **agent_stack_sz** to at least 64 4-KB pages. Complex XML schemas might require the value of **agent_stack_sz** to be much higher during schema registration or during XML document validation.

This limit is sufficient for most database operations.

Notes

- Agent stack memory does not count towards instance memory usage.
- While **agent_stack_sz** can be configured with a high stack space allocation for maximum usage, on average, only a small amount of allocated stack space is used by a thread. It is only this smaller amount which requires system memory.
- On HP-UX operating systems, thread stack space requires a swap reservation. The approximate total swap requirement is equal to:

Peak number of threads * **agent_stack_sz**

The value of **agent_stack_sz** in this calculation is rounded up to the next highest power-of-2 value.

agentpri - Priority of agents

This parameter controls the priority given to all agents and to other database manager instance processes and threads by the operating system scheduler. This priority determines how CPU time is allocated to the database manager processes, agents, and threads relative to other processes and threads running on the machine.

Important: The **agentpri** database manager configuration is deprecated since Version 9.5. It can still be used in pre-Version 9.5 data servers and clients. Also, agent priority for the WLM service class is discontinued in Version 10.5. Start to use the WLM dispatcher capability instead of agent priority. For more information, see “Agent priority of service classes is discontinued” in *What’s New for Db2 Version 10.5*.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

AIX -1 (system) [41 - 125]

Other UNIX

-1 (system) [41 - 128]

Windows

-1 (system) [0 - 6]

Solaris

-1 (system) [0 - 59]

Linux -1 (system) [1 - 99]

HP-UX

-1 (system) [0 - 31]

When the parameter is set to -1 or system, no special action is taken and the database manager is scheduled in the normal way that the operating system schedules all processes and threads. When the parameter is set to a value other than -1 or system, the database manager will create its processes and threads with

a static priority set to the value of the parameter. Therefore, this parameter allows you to control the priority with which the database manager processes and threads (in a partitioned database environment, this also includes coordinating and subagents, the parallel system controllers, and the FCM daemons) will execute on your machine.

You can use this parameter to increase database manager throughput. The values for setting this parameter are dependent on the operating system on which the database manager is running. For example, in a Linux or UNIX environment, numerically low values yield high priorities. When the parameter is set to a value between 41 and 125, the database manager creates its agents with a UNIX static priority set to the value of the parameter. This is important in Linux and UNIX environments because numerically low values yield high priorities for the database manager, but other processes (including applications and users) might experience delays because they cannot obtain enough CPU time. You should balance the setting of this parameter with the other activity expected on the machine.

Restrictions:

- If you set this parameter to a non-default value on Linux and UNIX platforms, you cannot use the governor to alter agent priorities.
- On the Solaris operating system, you should not change the default value (-1). Changing the default value sets the priority of Db2 processes to real-time, which can monopolize all available resources on the system.

Recommendation: The default value should be used initially. This value provides a good compromise between response time to other users/applications and database manager throughput.

If database performance is a concern, you can use benchmarking techniques to determine the optimum setting for this parameter. You should take care when increasing the priority of the database manager because performance of other user processes can be severely degraded, especially when the CPU utilization is very high. Increasing the priority of the database manager processes and threads can have significant performance benefits.

alt_diagpath - Alternate diagnostic data directory path

This parameter allows you to specify the fully qualified alternate path for Db2 diagnostic information that is used when the primary diagnostic data path, **diagpath**, is unavailable.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

Null [any valid path name, , `"pathname $h"`, `"pathname $h/trailing-dir"`, `"pathname $n"`,¹ `"pathname $n/trailing-dir"`, `"pathname $m"`, `"pathname $m/trailing-dir"`, `"pathname hn"`,² `"pathname hn/trailing-dir"`, `"pathname hm"`, or `"pathname hm/trailing-dir"`]

Symbols

pathname

A directory path to use when the primary diagnostic data directory path is unavailable

\$h Resolves to `HOST_hostname`

Note: Starting in Version 10, in Db2 pureScale environments, `$h` refers to the member's home host.

\$n Resolves to `NODENumber`

\$m Resolves to `DIAG_number`. Note that `DIAG_number` is used regardless of whether it refers to a database partition, a CF, or a member.

/trailing-dir

A single directory, or a directory and sub-directory to trail `$h` or `$n`

The following values are available:

- `"pathname $h"`
- `"pathname $h/trailing-dir"`
- `"pathname $n"`
- `"pathname $n/trailing-dir"`
- `"pathname $m"`
- `"pathname $m/trailing-dir"`
- `"pathname hn"`
- `"pathname hn/trailing-dir"`
- `"pathname hm"`
- `"pathname hm/trailing-dir"`

The alternate diagnostic data directory can contain the same diagnostic data as the primary diagnostic data directory set with the **diagpath** parameter. When **alt_diagpath** is set and the primary diagnostic data directory becomes unavailable, diagnostic logging continues in the alternate diagnostic data directory path specified, then resumes in its original location when the primary diagnostic path becomes available again. If this parameter is null and the primary diagnostic data directory specified by the **diagpath** parameter is unavailable, no further diagnostic information is written until the primary diagnostic path becomes available again. For improved resilience, set the alternate diagnostic data directory to point to a different file system than the primary diagnostic data directory.

Starting in Version 10.1, the alternate diagnostic data directory path writes to private `db2diag.log` for each member and CF, by default. To revert to the behavior of previous releases, in which the diagnostic data is written to the same directory, specify the **alt_diagpath** with a *pathname* and no token (`$h`, `$n`, or `$m`).

1. `$n` is deprecated and might be removed in a future release.

2. `hn` is deprecated and might be removed in a future release.

Note:

- To avoid the operating system shell interpreting the \$ sign on some Linux and UNIX systems, a single quote must be placed outside of the double quote, as shown in the syntax.
- In the CLP interactive mode, or if the command is read and executed from an input file, the double quote is not required.
- \$h, \$m, and \$n are case insensitive.
- The dynamic behavior for **alt_diagpath** does not extend to all processes.
- The **db2sysc** Db2 server process can detect dynamic changes, for example, when you issue the **UPDATE DATABASE MANAGER CONFIGURATION** command over an instance attachment.
- When Db2 client and application processes start, they use the **alt_diagpath** configuration parameter setting and do not detect any dynamic changes.
- On UNIX systems, if both **diagpath** and **alt_diagpath** are not available, the db2 diagnostic message is dumped to the syslog file.
- There is no default directory for **alt_diagpath** configuration parameter.
- The **alt_diagpath** and **diagpath** configuration parameters are exclusive to each other. They cannot be set to same directory path.
- If **alt_diagpath** (or **diagpath**) is unavailable that means diagnostic data dumping failed due to an error, such as: The directory was deleted, a disk error, disk is lost, network problems, file permission error, or disk is full.
- In Db2 pureScale environments, the CF diagnostic logs are placed into the **alt_diagpath** directory when the **cf_diagpath** directory is not available.

alternate_auth_enc - Alternate encryption algorithm for incoming connections at server configuration parameter

This configuration parameter specifies the alternate encryption algorithm used to encrypt the user IDs and passwords submitted to a Db2 database server for authentication. Specifically, this parameter affects the encryption algorithm when the authentication method negotiated between the Db2 client and the Db2 database server is SERVER_ENCRYPT.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

NOT_SPECIFIED [AES_CMP; AES_ONLY]

The user ID and password submitted for authentication on the Db2 database server are encrypted when the authentication method negotiated between the Db2 client and the Db2 server is SERVER_ENCRYPT. The authentication method negotiated depends on the authentication type setting on the server and the authentication type requested by the client. The choice of the encryption algorithm used to encrypt the user ID and password depends on the setting of the **alternate_auth_enc** database manager configuration parameter. It can be either DES or AES depending on this setting.

When the default (NOT_SPECIFIED) value is used, the database server accepts the encryption algorithm that the client proposes.

When **alternate_auth_enc** is set to AES_ONLY, the database server will only accept connections that use AES encryption. If the client does not support AES encryption, then the connection is rejected.

When **alternate_auth_enc** is set to AES_CMP, the database server will accept user IDs and passwords that are encrypted using either AES or DES, but it will negotiate for AES if the client supports AES encryption.

Note: You cannot set **alternate_auth_enc** to AES_CMP or AES_ONLY if **authentication** is set to DATA_ENCRYPT.

aslheapsz - Application support layer heap size

The application support layer heap represents a communication buffer between the local application and its associated agent. This buffer is allocated as shared memory by each database manager agent that is started.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

15 [1 - 524 288]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

When the database manager agent process is started for the local application

When freed

When the database manager agent process is terminated

If the request to the database manager, or its associated reply, do not fit into the buffer they will be split into two or more send-and-receive pairs. The size of this buffer should be set to handle the majority of requests using a single send-and-receive pair. The size of the request is based on the storage required to hold:

- The input SQLDA
- All of the associated data in the SQLVARs
- The output SQLDA
- Other fields which do not generally exceed 250 bytes.

In addition to this communication buffer, this parameter is also used for two other purposes:

- It is used to determine the I/O block size when a blocking cursor is opened. This memory for blocked cursors is allocated out of the application's private address space, so you should determine the optimal amount of private memory to allocate for each application program. If the Data Server Runtime Client cannot allocate space for a blocking cursor out of an application's private memory, a non-blocking cursor will be opened.
- It is used to determine the communication size between agents and **db2fmp** processes. (A **db2fmp** process can be a user-defined function or a fenced stored procedure.) The number of bytes is allocated from shared memory for each **db2fmp** process or thread that is active on the system.

The data sent from the local application is received by the database manager into a set of contiguous memory allocated from the query heap. The **as1heapsz** parameter is used to determine the initial size of the query heap (for both local and remote clients). The maximum size of the query heap is defined by the **query_heap_sz** parameter.

Recommendation: If your application's requests are generally small and the application is running on a memory constrained system, you might want to reduce the value of this parameter. If your queries are generally very large, requiring more than one send and receive request, and your system is not constrained by memory, you might want to increase the value of this parameter.

Use the following formula to calculate a minimum number of pages for **as1heapsz**:

```
as1heapsz >= ( sizeof(input SQLDA)
               + sizeof(each input SQLVAR)
               + sizeof(output SQLDA)
               + 250 ) / 4096
```

where `sizeof(x)` is the size of `x` in bytes that calculates the number of pages of a given input or output value.

You should also consider the effect of this parameter on the number and potential size of blocking cursors. Large row blocks might yield better performance if the number or size of rows being transferred is large (for example, if the amount of data is greater than 4096 bytes). However, there is a trade-off in that larger record blocks increase the size of the working set memory for each connection.

Larger record blocks might also cause more data than required to be block fetched by the application. You can control the number of fetch requests using the **OPTIMIZE FOR** clause on the **SELECT** statement in your application.

audit_buf_sz - Audit buffer size

This parameter specifies the size of the buffers used when you audit the Db2 instance.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
0 [0 - 65 000]

Unit of measure
Pages (4 KB)

When allocated
Two buffers are allocated from the DBMS memory set when the Db2 instance is started. Two buffers are allocated from the database heap for each database that is activated.

When freed
Instance-level audit buffers are freed when Db2 is stopped. Database-level audit buffers are freed when the database is deactivated.

If you set the value of **audit_buf_sz** configuration parameter to zero (0), audit buffers are not used. If you set the value greater than zero (0), space is allocated for the audit buffers where the audit records are placed when they are generated by the audit facility. The amount of space allocated for each audit buffer is the value of **audit_buf_sz** times the page size of 4KB. At regular time intervals, or when an audit buffer is full, the **db2auditd** audit daemon process flushes the audit buffer to disk.

Audit buffers cannot be allocated dynamically. You must restart the Db2 instance before the new value for this parameter takes effect.

If you change **audit_buf_sz** configuration parameter to a value larger than zero (0), the audit facility writes records to disk asynchronously compared to the execution of the statements generating the audit records, which improves the performance of the Db2 instance. The value of zero (0) means the audit facility writes records to disk synchronously with the execution of the statements generating the audit records. The synchronous operation during auditing decreases the performance of applications running in Db2.

authentication - Authentication type

This parameter specifies and determines how and where authentication of a user takes place.

Configuration type
Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DATA_ENCRYPT; DATA_ENCRYPT_CMP; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

If the value of the **authentication** parameter is `SERVER`, the user ID and password are sent from the client to the server so that authentication can take place on the server. The `SERVER_ENCRYPT` value provides the same behavior as the `SERVER` value, except that any user IDs and passwords that are sent over the network must be encrypted.

Starting with the Db2 Cancun Release (Db2 10.5.0.4), in order for the Db2 server to not accept `CLEAR_TEXT_PASSWORD_SECURITY` security mechanism when the authentication type is `SERVER_ENCRYPT`, set the Db2 **DB2AUTH** registry variable to `JCC_ENFORCE_SECMEC` at the server.

Starting with Db2 11.1.3.3, set the Db2 **DB2AUTH** registry variable to `JCC_NOENFORCE_SECMEC_MSG` at the server in order to print the following warning message in the `db2diag.log` (for each connection) when Db2 accepts the `CLEAR_TEXT_PASSWORD_SECURITY` security mechanism from Java clients when the authentication type is `SERVER_ENCRYPT`: Connection accepted as `SERVER` (`JCC_NOENFORCE_SECMEC_MSG`).

Another new value is introduced in Db2 11.1.3.3, `JCC_NOENFORCE_SECMEC_NOMSG`, to represent the default configuration behavior for the `SERVER_ENCRYPT` authentication type which is to accept the security mechanisms but not print any warning to the `db2diag.log`.

To use AES, install the "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy" files from Oracle.

For a standards compliance (defined in the "Standards compliance" topic) configuration, `SERVER` is the only supported value.

A value of `DATA_ENCRYPT` means the server accepts encrypted `SERVER` authentication schemes and the encryption of user data. The authentication works the same way as `SERVER_ENCRYPT`.

The following user data is encrypted when you use the `DATA_ENCRYPT` authentication type:

- SQL and XQuery statements
- SQL program variable data
- Output data from the server processing an SQL or XQuery statement and including a description of the data
- Some or all of the answer set data resulting from a query
- Large object (LOB) streaming
- SQLDA descriptors

A value of `DATA_ENCRYPT_CMP` means the server accepts encrypted `SERVER` authentication schemes and the encryption of user data. In addition, this authentication type provides compatibility with earlier products that do not support the `DATA_ENCRYPT` authentication type. These products are permitted to connect with the `SERVER_ENCRYPT` authentication type, without encrypting user data. Products supporting the new authentication type must use it. This authentication type is valid only in the server's database manager configuration file and is not valid for the **CATALOG DATABASE** command.

You cannot set the **authentication** parameter to `DATA_ENCRYPT` if you set the **alternate_auth_enc** parameter to `AES_CMP` or `AES_ONLY`.

A value of CLIENT indicates that all authentication takes place at the client. No authentication needs to be performed at the server.

A value of KERBEROS means that authentication is performed at a Kerberos server by using the Kerberos security protocol. With an authentication type of KRB_SERVER_ENCRYPT at the server and clients that support the Kerberos security system, the effective system authentication type is KERBEROS. If the clients do not support the Kerberos security system, the system authentication type is effectively equivalent to SERVER_ENCRYPT.

A value of GSSPLUGIN means that authentication is performed using an external GSSAPI-based security mechanism. With an authentication type of GSS_SERVER_ENCRYPT at the server and clients that support the GSSPLUGIN security mechanism, the effective system authentication type is GSSPLUGIN if the clients support one of the server's plug-ins. If the clients do not support the GSSPLUGIN security mechanism, the system authentication type is effectively equivalent to SERVER_ENCRYPT.

Recommendation: Typically, the default value (SERVER) is adequate for local clients. If remote clients are connecting to the database server, SERVER_ENCRYPT is the suggested value to protect the user ID and password.

Note: SERVER_ENCRYPT will only protect the userid-password pair. If you are concerned about other plain-text data being flown, then use SSL.

cf_diaglevel - diagnostic error capture level configuration parameter for the CF

This parameter specifies the type of diagnostic errors that will be recorded in the cfdiag*.log files

Configuration type
Database Manager

Applies to
• Db2 pureScale

Parameter type
Configurable offline

Propagation class
Immediate

Default [range]
2 [1 - 4]

Valid values for this parameter are:

- 0 - No diagnostic data captured
- 1 - Severe errors only
- 2 - All errors
- 3 - All errors and warnings
- 4 - All errors, warnings and informational messages

The **cf_diagpath** configuration parameter is used to specify the directory that will contain the diagnostic message file that will be generated, based on the value of the **cf_diaglevel** parameter

The CF log file location will be determined in the following order:

- **cf_diagpath** (DBM configuration parameter)
- **diagpath** (DBM configuration parameter)
- **DB2PATH/db2dump**

cf_diagpath - diagnostic data directory path configuration parameter for the CF

This parameter allows you to specify the fully qualified path for the diagnostic information file for the CF

Configuration type

Database Manager

Applies to

- Db2 pureScale

Parameter type

Configurable offline

Propagation class

Immediate

Default [range]

INSTHOME/sqlib/db2dump/ \$m [any valid path name]

Starting in Version 10.1, the cf diagnostic data directory path writes to a private db2diag.log for each CF by default. To revert to the behavior of previous releases, in which the diagnostic data for the CF is written to the same directory, specify the **cf_diagpath** with a *pathname* and no token.

Note: When the **cf_diagpath** directory is not available, but the **alt_diagpath** is available, the CF diagnostic logs are placed into the **alt_diagpath** directory.

Each CF log file name has the following format:

```
cfdiag-YYYYMMDDhhmmssuuuuu.<cf#>.log
```

However, there is also a single static CF diagnostic log name that always points to the most current CF diagnostic logging file and has the following format:

```
cfdiag.<cf#>.log
```

For example, if you listed all of the CF log files, you would see something similar to the following:

```
$ ls -la cfdiag*
lrwxrwxrwx 1 db2inst1 pxdxb2 35 2011-02-09 15:07
  cfdiag.128.log -> cfdiag-20110209150733000049.128.log
lrwxrwxrwx 1 db2inst1 pxdxb2 35 2011-02-09 15:10
  cfdiag.129.log -> cfdiag-20110209151021000040.129.log
-rw-r-xr-- 1 db2inst1 pxdxb2 1271 2011-02-09 15:07
  cfdiag-20110209150733000049.128.log
-rw-r-xr-- 1 db2inst1 pxdxb2 1271 2011-02-09 15:07
  cfdiag-20110209150740000082.129.log
-rw-r-xr-- 1 db2inst1 pxdxb2 1274 2011-02-09 15:10
  cfdiag-20110209151021000040.129.log
```

In this example, cfdiag.128.log and cfdiag.129.log are symbolic links to the latest versions of the otherwise time stamped log files.

The CF log file looks similar to the db2diag.log file.

cf_mem_sz - CF memory configuration parameter

This parameter controls the total memory that is used by the cluster caching facility, also known as CF.

Configuration type

Database Manager

Applies to

Applies to a Db2 pureScale instance only.

- Database server with local and remote clients

Parameter type

Configurable offline

Default [range]

AUTOMATIC [32768 - 4 294 967 295]

Unit of measure

Pages (4 KB)

When allocated

When the CF is started

When freed

When the CF is stopped

When the default setting AUTOMATIC is applied, the amount of cluster caching facility memory is determined by querying the total memory that is available on the CF server. The parameter is then set to either an appropriate percentage of the total memory on the CF server, or the amount of free memory on the machine, whichever value is less. An appropriate percentage is typically 70% to 90% of the total memory available on the CF. Factors for consideration during AUTOMATIC computation also include:

- If the CF and any Db2 members coexist
- If there are multiple instances on the same host

cf_num_conns - Number of CF connections per member per CF configuration parameter

This parameter controls the initial size of the cluster caching facility (CF) connection pool.

Configuration type

Database Manager

Parameter type

Configurable online

Default [range]

AUTOMATIC [4 - 256]

When allocated

When Db2 is started

When freed

When Db2 is stopped

When you set the **cf_num_conns** parameter to AUTOMATIC (the default), the Db2 database manager creates an initial number of CF connections for each member with each CF at start time. This initial number is based on the number of worker threads, (See: "cf_num_workers - Number of worker threads configuration

parameter”), number of connections per worker thread, and the number of members in the cluster. The actual maximum value for the **cf_num_conns** parameter is calculated automatically by Db2 at member startup, based on those parameters.

If the **cf_num_conns** parameter is set to AUTOMATIC and then raised to a value higher than the current value, with an instance attachment, new CF connections are created. However, if you set the **cf_num_conns** parameter to a value lower than the current value, CF connections are not closed.

When the **cf_num_conns** parameter is set to AUTOMATIC, any dropped connections caused by a port failure (or other network problem) are reestablished evenly on the remaining ports. When a failed port comes back online, connections are rebalanced as required.

When you set the **cf_num_conns** parameter to a fixed numeric value, the Db2 database manager creates exactly that number of CF connections for each member with each CF at start time. There is no automatic growing or shrinking done by Db2 database manager.

If the **cf_num_conns** parameter is set to a fixed value, and then the value is increased or decreased, with an instance attachment, then the number of CF connections is increased or decreased immediately, according to the new value.

When the **cf_num_conns** parameter is set to a fixed value, a port failure does not increase the number of connections on the remaining ports. The number of connections made to each port remains at the number found by dividing the **cf_num_conns** parameter by the number of ports. That number does not change, even if some ports are offline.

cf_num_workers - Number of worker threads configuration parameter

The **cf_num_workers** parameter specifies the total number of worker threads on the cluster caching facility (CF). Worker threads are distributed among the communication adapter ports to balance the number of worker threads servicing requests on each interface.

Configuration type
Database Manager

Applies to
Applies to a Db2 pureScale instance only.

Parameter type
Configurable offline

Default [range]
AUTOMATIC [1 - 31]

When set to the default (AUTOMATIC), the parameter value is configured to be one less than the number of available processors on the CF. The available CPUs are equally divided among the instances before one processor is subtracted from the resulting value for each instance. If there are coexisting members on the CF host, the number of worker threads on the CF is further divided by the total number of CF and members on a host.

If there is only one processor on the CF server machine, this value is set to 1.

The total number of CF worker threads is shown in the `cfdiag.log` file.

To calculate the number of worker threads assigned to each cluster interconnect interface, divide the total number of CF work threads by the number of communication adapter ports defined for the CF.

Number of workers assigned to each interface =
 $(\text{cf_num_workers}) / (\text{number of interfaces})$

When the **cf_num_workers** parameter is set to AUTOMATIC, the database manager configures the number of worker threads so that it is equally divisible by the number of communication adapter ports configured for the CF.

For example, on a CF server machine with 7 available processors, and 2 communication adapter ports, the value for the AUTOMATIC setting is:

Total CF worker threads =
 $(7 \text{ processors}) - (1 \text{ processor to prevent using all processors}) = 6$
Thus, number of workers connected to each cluster interconnect interface =
 $6 / 2 = 3$

If you set the **cf_num_workers** parameter manually, set the value to be equal or greater than the number of communication adapter ports so that there is at least one worker thread for each interface. If there is an insufficient number of worker threads to cover all interfaces, an alert is logged for the CF, which will fail to start. The parameter value must be changed to address the problem.

In a Db2 pureScale environment that uses sockets as the method of communication the AUTOMATIC value is configured differently. When sockets are used and the **cf_num_workers** is set to default (AUTOMATIC), the parameter value is configured to be four times the number of available processors on the instance.

Restrictions:

If you are running InfiniBand or uDAPL, do not set this value higher than the number of processors on the CF server machine. Each worker thread operates on a processor, waiting for uDAPL communication. Performance is affected if the number of worker threads exceeds the number of processors.

cf_transport_method - Network transport method

In Db2 pureScale environments, the **cf_transport_method** configuration parameter controls what method is used for communication between Db2 members and the cluster caching facility (CF).

Configuration type

Database manager

Parameter type

Configurable offline

Default [range]

RDMA [RDMA, TCP]

When **cf_transport_method** is set to RDMA, Db2 members must communicate with the CF by using remote direct memory access (RDMA). To use an RDMA protocol network, the hardware configurations that you must use are an InfiniBand network with the appropriate InfiniBand hardware installed or an Ethernet network that uses a RoCE adapter card.

When the **cf_transport_method** parameter is set to TCP, Db2 members communicate with the CF by using a TCP/IP protocol network.

catalog_noauth - Cataloging allowed without authority

This parameter specifies whether users are able to catalog and uncatalog databases and nodes, or DCS and ODBC directories, without SYSADM authority.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Database server with local and remote clients

NO [NO (0) - YES (1)]

Client; Database server with local clients

YES [NO (0) - YES (1)]

The default value (0) for this parameter indicates that SYSADM authority is required. When this parameter is set to 1 (yes), SYSADM authority is not required.

clnt_krb_plugin - Client Kerberos plug-in

This parameter specifies the name of the default Kerberos plug-in library to be used for client-side authentication and local authorization.

Configuration type

Database manager

Applies to

- Client
- Database server with local and remote clients
- Database server with local clients only
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null on Linux and UNIX operating systems and IBMkrb5 on Windows operating systems [any valid string]

The plug-in is used when the client is authenticated using Kerberos authentication or when local authorization is performed and the authentication type in the database manager configuration is set to KERBEROS.

clnt_pw_plugin - Client userid-password plug-in

This parameter specifies the name of the userid-password plug-in library to be used for client-side authentication and local authorization.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid string]

By default, the value is null and the Db2 supplied userid-password plug-in library is used. The plug-in is used when the client is authenticated using CLIENT authentication, or when local authorization is performed and the **authentication** in the DBM CFG is CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT, or DATA_ENCRYPT_CMP. For non-root installations, if the Db2 userid and password plug-in library is used, the **db2rfe** command must be run before using your Db2 database product.

cluster_mgr - Cluster manager name

This parameter enables the database manager to communicate incremental cluster configuration changes to the specified cluster manager.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Multi-partitioned database server with local and remote clients

Parameter type

Informational

Default

- In a Db2 pureScale environment: TSA; otherwise, no default

Valid values

- TSA

This parameter is set automatically when you are installing the Db2 pureScale Feature or if you are using the Db2 High Availability Instance Configuration Utility (**db2haicu**) to configure your cluster for high availability.

comm_bandwidth - Communications bandwidth

This parameter helps the query optimizer determine access paths by indicating the bandwidth between database partition servers.

Configuration type

Database manager

Applies to

Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Statement boundary

Default [range]

-1 [-1, 0.1 - 100000]

A value of -1 causes the parameter value to be reset to the default. The default value is calculated based on the speed of the underlying communications adapter. A value of 100 can be expected for systems using Gigabit Ethernet.

Unit of measure

Megabytes per second

The value calculated for the communications bandwidth, in megabytes per second, is used by the query optimizer to estimate the cost of performing certain operations between the database partition servers of a partitioned database system. The optimizer does not model the cost of communications between a client and a server, so this parameter should reflect only the nominal bandwidth between the database partition servers, if any.

You can explicitly set this value to model a production environment on your test system or to assess the impact of upgrading hardware.

Recommendation: You should only adjust this parameter if you want to model a different environment.

The communications bandwidth is used by the optimizer in determining access paths. You should consider rebinding applications (using the **REBIND PACKAGE** command) after changing this parameter.

comm_exit_list - Communication exit library list

This parameter specifies the list of communication exit libraries that the database manager uses. A communication exit library is a dynamically loaded library that vendor applications use to examine communication buffers and the database manager runtime environment.

Configuration type

Database Manager

Applies To

- Database server with local and remote clients.
- Database server with local clients.
- Partitioned database server with local and remote clients.

Parameter type

Configurable

Default [range]

Null [any valid string]

By default, the value is null. When the value is not null, it is taken to be a comma separate list of communication exit libraries that the database manager loads. Each library must be separated by a comma, with no spaces either before or after the comma.

Each name can be up to 32 bytes long. The name must not include the extension like .so or .a. The total length of the parameter must not be more than 128 bytes.

conn_elapse - Connection elapse time

This parameter specifies the number of seconds within which a network connection is to be established between Db2 members.

Configuration type

Database manager

Applies to

Db2 pureScale server (with more than one Db2 member)

Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

10 [0-100]

Unit of measure

Seconds

If the attempt to connect succeeds within the time specified by this parameter, communications are established. If it fails, another attempt is made to establish communications. If the connection is attempted the number of times specified by the **max_connretries** parameter and always times out, an error is issued.

cpuspeed - CPU speed

This parameter reflects the CPU speed of the machine(s) the database is installed on.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Statement boundary

Default [range]

-1 [1×10^{-10} - 1] A value of -1 will cause the parameter value to be reset based on the running of the measurement program.

Unit of measure

Milliseconds

This program is executed if benchmark results are not available if the data for the IBM RS/6000® model 530H is not found in the file, or if the data for your machine is not found in the file.

You can explicitly set this value to model a production environment on your test system or to assess the impact of upgrading hardware. By setting it to -1, **cpuspeed** will be re-computed.

Recommendation: You should only adjust this parameter if you want to model a different environment.

The CPU speed is used by the optimizer in determining access paths. You should consider rebinding applications (using the **REBIND PACKAGE** command) after changing this parameter.

cur_eff_arch_level - Current effective architecture level configuration parameter

This parameter displays the current effective architecture level (CEAL) at which the instance is operating.

Configuration type

Database manager

Applies to

- All editions that support Db2 pureScale Feature

Parameter type

Informational

You can update a Db2 pureScale instance to a higher code level within a release by performing an offline fix pack update or an online fix pack update to keep the database online.

During an offline fix pack update or an online fix pack update of a Db2 pureScale instance, you can have members that have a different architecture level than the CEAL. Use the **cur_eff_arch_level** to display the CEAL of the instance. Use the **db2pd** command with the **-ruStatus** parameter to display the architecture level in a member.

cur_eff_code_level - Current effective code level configuration parameter

This parameter displays the current effective code level (CECL) at which the instance is operating.

Configuration type

Database manager

Applies to

- All editions that support Db2 pureScale Feature

Parameter type

Informational

You can update a Db2 pureScale instance to a higher code level within a release by performing an offline fix pack update or an online fix pack update to keep the database online.

During an offline fix pack update or an online fix pack update of a Db2 pureScale instance, you can have members that have a different code level than the CECL. Use the **cur_eff_code_level** to display the CECL of the instance. Use the **db2pd** command with the **-ruStatus** parameter to display the code level in a member.

date_compat - Date compatibility database configuration parameter

This parameter indicates whether the DATE compatibility semantics associated with the TIMESTAMP(0) data type are applied to the connected database.

Configuration type

Database

Parameter type

Informational

The value is determined at database creation time, and is based on the setting of the **DB2_COMPATIBILITY_VECTOR** registry variable for DATE data type. The value cannot be changed.

dft_account_str - Default charge-back account

This parameter acts as the default suffix of accounting identifiers.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Null [any valid string]

With each application connect request, an accounting identifier consisting of a Db2 Connect-generated prefix and the user supplied suffix is sent from the application requester to a DRDA application server. This accounting information provides a mechanism for system administrators to associate resource usage with each user access.

Note: This parameter is only applicable to Db2 Connect.

The suffix is supplied by the application program calling the `sqlsact()` API or the user setting the environment variable **DB2ACCOUNT**. If a suffix is not supplied by either the API or environment variable, Db2 Connect uses the value of this parameter as the default suffix value. This parameter is particularly useful for earlier database clients (anything before version 2) that do not have the capability to forward an accounting string to Db2 Connect.

Recommendation: Set this accounting string using the following possible values:

- Alphabetics (A through Z)
- Numerics (0 through 9)
- Underscore (_).

dft_monswitches - Default database system monitor switches

This parameter allows you to set a number of switches which are each internally represented by a bit of the parameter.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Note: The change takes effect immediately if you explicitly ATTACH to the instance before modifying the dft_mon_xxxx switch settings. Otherwise the setting takes effect the next time the instance is restarted.

Default

All switches turned off, except dft_mon_timestamp, which is turned on by default

The parameter is unique in that you can update each of these switches independently by setting the following parameters:

dft_mon_uow

Default value of the snapshot monitor's unit of work (UOW) switch

dft_mon_stmt

Default value of the snapshot monitor's statement switch

dft_mon_table

Default value of the snapshot monitor's table switch

dft_mon_bufpool

Default value of the snapshot monitor's buffer pool switch

dft_mon_lock

Default value of the snapshot monitor's lock switch

dft_mon_sort

Default value of the snapshot monitor's sort switch

dft_mon_timestamp

Default value of the snapshot monitor's timestamp switch

Recommendation: Any switch (except dft_mon_timestamp) that is turned ON instructs the database manager to collect monitor data related to that switch. Collecting additional monitor data increases the processing time of the database manager which can impact system performance. Turning the dft_mon_timestamp switch OFF becomes important as CPU utilization approaches 100%. When this occurs, the CPU time required for issuing timestamps increases dramatically.

Furthermore, if the timestamp switch is turned OFF, the overall cost of other data under monitor switch control is greatly reduced.

All monitoring applications inherit these default switch settings when the application issues its first monitoring request (for example, setting a switch, activating the event monitor, taking a snapshot). You should turn on a switch in the configuration file only if you want to collect data starting from the moment the database manager is started. (Otherwise, each monitoring application can set its own switches and the data it collects becomes relative to the time its switches are set.)

dftdbpath - Default database path

This parameter contains the default file path used to create databases under the database manager. If no path is specified when a database is created, the database is created under the path specified by the **dftdbpath** parameter.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

UNIX Home directory of instance owner [any existing path]

Windows

Drive on which the Db2 database system is installed [any existing path]

In a partitioned database environment, you should ensure that the path on which the database is being created is not an NFS-mounted path (on Linux and UNIX operating systems), or a network drive (in a Windows environment). The specified path must physically exist on each database partition server. To avoid confusion, it is best to specify a path that is locally mounted on each database partition server. The maximum length of the path is 205 characters. The system appends the database partition name to the end of the path.

Given that databases can grow to a large size and that many users could be creating databases (depending on your environment and intentions), it is often convenient to be able to have all databases created and stored in a specified location. It is also good to be able to isolate databases from other applications and data both for integrity reasons and for ease of backup and recovery.

For Linux and UNIX environments, the length of the **dftdbpath** name cannot exceed 215 characters and must be a valid, absolute, path name. For Windows, the **dftdbpath** can be a drive letter, optionally followed by a colon.

Recommendation: If possible, put high volume databases on a different disk than other frequently accessed data, such as the operating system files and the database logs.

diaglevel - Diagnostic error capture level

This parameter specifies the type of diagnostic errors that will be recorded in the db2diag log file.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

3 [0 — 4]

Valid values for this parameter are:

- **0** – Starting in Fix Pack 1, only administration notification messages are captured on the client side. Critical errors, event messages, and administration notification messages are still captured on the server side.
- **1** – Only severe errors, critical errors, event messages, and administration notification messages are captured.
- **2** – All errors, event messages, and administration notification messages are captured.
- **3** – All errors, warnings, event messages, and administration notification messages are captured. Some key informational messages might be captured at this level as well.
- **4** – All errors, warnings, informational messages, event messages, and administration notification messages are captured.

The **diagpath** configuration parameter is used to specify the directory that will contain the error file, alert log file, and any dump files that might be generated, based on the value of the **diaglevel** parameter.

Usage notes

- The dynamic behaviour for **diaglevel** does not extend to all processes.
- The **db2sysc** Db2 server process can detect dynamic changes, for example, when you issue the **UPDATE DATABASE MANAGER CONFIGURATION** command over an instance attachment.
- When Db2 client and application processes start, they use the **diaglevel** configuration parameter setting and do not detect any dynamic changes.
- To help resolve a problem, you can increase the value of this parameter to gather additional problem determination data.

diagpath - Diagnostic data directory path

This parameter allows you to specify the fully qualified primary path for Db2 diagnostic information.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

`$INSTHOME/sql11ib/db2dump/ $m` [any valid path name (see the following section for resolution variables options)]

`$INSTHOME/sql11ib/db2dump` for instance configuration in a single member environment

Symbols

pathname

A directory path to use instead of the default diagnostic data directory

\$h Resolves to `HOST_hostname`.

Note: Starting in Version 10, in Db2 pureScale environments, `$h` refers to the member's home host.

\$n Resolves to `NODENumber`

\$m Resolves to `DIAGnumber`. Note that `DIAGnumber` is used regardless of whether it refers to a database partition, a CF, or a member.

/trailing-dir

A single directory, or a directory and sub-directory to trail `$h` or `$n`

The following values are available:

- `''$h''`
- `''$h/trailing-dir''`
- `''pathname $h''`
- `''pathname $h/trailing-dir''`
- `''$n''3`
- `''$n/trailing-dir''`
- `''pathname $n''`
- `''pathname $n/trailing-dir''`
- `''$m''`
- `''$m/trailing-dir''`

3. `$n` is deprecated and might be removed in a future release.

- `"pathname $m"`
- `"pathname $m/trailing-dir"`
- `"hn"4`
- `"hn/trailing-dir"`
- `"pathname hn"`
- `"pathname hn/trailing-dir"`
- `"hm"`
- `"hm/trailing-dir"`
- `"pathname hm"`
- `"pathname hm/trailing-dir"`

The primary diagnostic data directory could possibly contain dump files, trap files, an error log, a notification file, an alert log file, and first occurrence data collection (FODC) packages, depending on your platform.

If this parameter is null, the diagnostic information will be written to a default diagnostic path directory string in one of the following directories or folders:

- In Windows environments:
 - The default location of user data files, for example, files under instance directories, varies from edition to edition of the Windows family of operating systems. Use the **DB2SET DB2INSTPROF** command to get the location of the instance directory. The file is in the instance subdirectory of the directory specified by the **DB2INSTPROF** registry variable.
- In Linux and UNIX environments: Information is written to `INSTHOME/sqllib/db2dump/ $m`, where `INSTHOME` is the home directory of the instance.

Starting in Version 10.1, the diagnostic data directory path writes to a private `db2diag` log file for each member and CF, by default. To revert to the behavior of previous releases, in which the diagnostic data is written to the same directory, specify the **diagpath** with a *pathname* and no token (`$h`, `$n`, or `$m`).

To split the diagnostic data directory path to collect diagnostic information per physical host, set the parameter to one of the following values:

- Split default diagnostic data directory path:
`db2 update dbm cfg using diagpath "$h"`

which creates a subdirectory under the default diagnostic data directory with the host name, as in the following:

`Default_diagpath/HOST_hostname`

- Split default diagnostic data directory path with a trailing directory:
`db2 update dbm cfg using diagpath "$h/trailing-dir"`

which creates a subdirectory under the default diagnostic data directory with the host name and a trailing directory, as in the following:

`Default_diagpath/HOST_hostname/trailing-dir`

- Split your own specified diagnostic data directory path (there is a blank space between *pathname* and `$h`):
`db2 update dbm cfg using diagpath "pathname $h"`

4. `hn` is deprecated and might be removed in a future release.

which creates a subdirectory under your own specified diagnostic data directory with the host name, as in the following:

```
pathname/HOST_ hostname
```

- Split your own specified diagnostic data directory path (there is a blank space between *pathname* and \$h) and with a trailing directory:

```
db2 update dbm cfg using diagpath "pathname $h/trailing-dir"
```

which creates a subdirectory under your own specified diagnostic data directory with the host name and a trailing directory, as in the following:

```
pathname/HOST_ hostname/trailing-dir
```

To split the diagnostic data directory path to collect diagnostic information per physical host and per database partition per physical host, set the parameter to one of the following values:

- Split default diagnostic data directory path:

```
db2 update dbm cfg using diagpath "$h$n"
```

which creates a subdirectory for each logical partition on the host under the default diagnostic data directory with the host name and the partition number, as in the following:

```
Default_diagpath/HOST_ hostname/NODEnumber
```

- Split default diagnostic data directory path with a trailing directory:

```
db2 update dbm cfg using diagpath "$h$n/trailing-dir"
```

which creates a subdirectory for each logical partition on the host under the default diagnostic data directory with the host name, the partition number, and a trailing directory, as in the following:

```
Default_diagpath/HOST_ hostname/NODEnumber/trailing-dir
```

- Split your own specified diagnostic data directory path (there is a blank space between *pathname* and \$h\$n):

```
db2 update dbm cfg using diagpath "pathname $h$n"
```

which creates a subdirectory for each logical partition on the host under your own specified diagnostic data directory with the host name and the partition number, as in the following:

```
pathname/HOST_ hostname/NODEnumber
```

For example, an AIX host, named boson, has 3 database partitions with node numbers 0, 1, and 2. An example of a list output for the directory is similar to the following:

```
usr1@boson /home/user1/db2dump->ls -R *  
HOST_boson:
```

```
HOST_boson:  
NODE0000 NODE0001 NODE0002
```

```
HOST_boson/NODE0000:  
db2diag.log db2eventlog.000 db2resync.log db2samp1_Import.msg events usr1.nfy
```

```
HOST_boson/NODE0000/events:  
db2optstats.0.log
```

```
HOST_boson/NODE0001:  
db2diag.log db2eventlog.001 db2resync.log usr1.nfy stmmlog
```

```
HOST_boson/NODE0001/stmmlog:  
stmm.0.log
```

```
HOST_boson/NODE0002:  
db2diag.log db2eventlog.002 db2resync.log usr1.nfy
```

- Split your own specified diagnostic data directory path (there is a blank space between *pathname* and *\$h\$n*) and with a trailing directory:

```
db2 update dbm cfg using diagpath "pathname $h$n/trailing-dir"
```

which creates a subdirectory for each logical partition on the host under your own specified diagnostic data directory with the host name, the partition number and a trailing directory, as in the following:

```
pathname/HOST_hostname/NODEnumber/trailing-dir
```

Note:

- To avoid the operating system shell interpreting the \$ sign on some Linux and UNIX systems, a single quote must be placed outside of the double quote, as shown in the syntax.
- In the CLP interactive mode, or if the command is read and executed from an input file, the double quote is not required.
- \$h, \$n, and \$m are case insensitive.
- The dynamic behavior for **diagpath** does not extend to all processes
- The **db2sysc** Db2 server process can detect dynamic changes, for example, when you issue the **UPDATE DATABASE MANAGER CONFIGURATION** command over an instance attachment.
- When Db2 client and application processes start, they use the **diagpath** configuration parameter setting and do not detect any dynamic changes.
- When the **diagpath** configuration parameter is set to a new directory, it is recommended that this new **diagpath** directory has the same permission as the default **diagpath** directory.

For Linux and UNIX environment, the preferred permission value on the new directory is 1777. To set the permission on the new directory, run the following command:

```
chmod 1777 <directory>
```

In Version 9.5 and later, the default value of **DB2INSTPROF** at the global level is stored at the new location shown previously. Other profile registry variables that specify the location of the runtime data files should query the value of **DB2INSTPROF**. The other variables include the following ones:

- **DB2CLIINIPATH**
- **DIAGPATH**
- **SPM_LOG_PATH**

Note: In Db2 Version 9.7 Fix Pack 4 and later fix packs, diagnostic logging can be made more resilient by setting an alternate diagnostic path in conjunction with the **diagpath** parameter. When **alt_diagpath** is set and the path specified by **diagpath** becomes unavailable, diagnostic logging continues in the alternate diagnostic data directory path specified, then resumes when the primary diagnostic path becomes available again.

diagsize - Rotating diagnostic and administration notification logs configuration parameter

This parameter helps control the maximum sizes of the diagnostic log and administration notification log files.

Configuration type

Database manager

Parameter type

Not configurable online

Default

0

Minimum value for specifying the size of rotating logs:

2

Maximum value for specifying the size of rotating logs:The amount of free space in the directory specified by `diagpath`**Unit of measure**

Megabytes

If the value of this parameter is 0, the default, there is only one diagnostic log file, called the `db2diag.log` file. There is also only one administration notification log file, called the `<instance>.nfy` file, which is used only on Linux and UNIX operating systems. The sizes of these files can increase indefinitely.

If you set the parameter to a non-zero value and restart the `<instance>`, a series of rotating diagnostic log files and a series of rotating administration notification log files are used. These files are called the `db2diag.n.log` and `<instance>.n.nfy` files, where `n` is an integer; `<instance>.n.nfy` files apply only to Linux and UNIX operating systems. The number of `db2diag.n.log` files and `<instance>.n.nfy` files cannot exceed 10 each. When the size of 10th file is full, the oldest file is deleted, and a new file is created.

For example, on Linux and UNIX operating systems the rotating log files under **diagpath** might look like the following output:

```
db2diag.14.log, db2diag.15.log, ... , db2diag.22.log, db2diag.23.log  
<instance>.0.nfy, <instance>.1.nfy..., <instance>.8.nfy, <instance>.9.nfy
```

If `db2diag.23.log` is full, `db2diag.14.log` will be deleted, `db2diag.24.log` will be created for `db2diag` logging

If `<instance>.9.nfy` is full, `<instance>.0.nfy` is deleted, `<instance>.10.nfy` will be created for administration notification logging.

Note that the messages are always logged to rotating log file with the largest index number `db2diag.largest n.log`, `<instance>.largest n.nfy`

The total size of the `db2diag.n.log` and `<instance>.n.nfy` files are determined by the value of the **diagsize** configuration parameter. By default, except on Windows operating systems, 90% of the value of **diagsize** is allocated to the `db2diag.n.log` files, and 10% of the value of **diagsize** is allocated to the `<instance>.n.nfy` files. For example, if you set **diagsize** to 1024 on a Linux or UNIX operating system, the total size of the `db2diag.n.log` files cannot exceed 921.6 MB, and the total size of the `<instance>.n.nfy` files cannot exceed 102.4 MB. On Windows operating systems, the total value of **diagsize** is allocated to the `db2diag.n.log` files. The size of each log file is determined by the total amount of space allocated to each type of log file divided by 10. For example, if the total size of the `db2diag.n.log` files cannot exceed 921.6 MB, the size of each `db2diag.n.log` file is 92.16 MB.

The maximum value that you specify for the **diagsize** configuration parameter cannot exceed the amount of free space in the directory that you specify for the **diagpath** configuration parameter. The diagnostic and administration notification log files are stored in this directory. To avoid losing information too quickly because of file rotation (the deletion of the oldest log file), set **diagsize** to a value that is greater than 50 MB but not more than 80% of the free space in the directory that you specify for **diagpath**.

For example,

- To set the **diagsize** to 1024 MB, that will switch to rotate logging behavior when Db2 gets restarted, use the following command:
`db2 update dbm cfg using diagsize 1024`
- To set the **diagsize** to 0 that will switch to default logging behavior when Db2 gets restarted, use the following command:
`db2 update dbm cfg using diagsize 0`

If you change the value of the **diagsize** configuration parameter and do not restart the instance, the instance and existing processes continue to use the old value. Any new processes that are created after the value is changed uses the new value. This might result in 2 log files being written to until the instance is restarted.

Note: Starting with Db2 Version 9.7 Fix Pack 1, if the **diagsize** configuration parameter is set to a non-zero value and the **diagpath** configuration parameter is set to split the diagnostic data into separate directories, then the non-zero value of the **diagsize** configuration parameter specifies the total size of the combination of all rotating administration notification log files and all rotating diagnostic log files contained within a given split diagnostic data directory. For example, if a system with 4 database partitions has **diagsize** set to 1 GB and **diagpath** set to "\$n" (split diagnostic data per database partition), the maximum total size of the combined notification and diagnostic logs can reach 4 GB (4 x 1 GB).

dir_cache - Directory cache support

This parameter determines whether the database, node and DCS directory files will be cached in memory.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Yes [Yes; No]

When allocated

- When an application issues its first connect, the application directory cache is allocated
- When a database manager instance is started (**db2start**), the server directory cache is allocated.

When freed

- When an the application process terminates, the application directory cache is freed
- When a database manager instance is stopped (**db2stop**), the server directory cache is freed.

The use of the directory cache reduces connect costs by eliminating directory file I/O and minimizing the directory searches required to retrieve directory information. There are two types of directory caches:

- An application directory cache that is allocated and used for each application process on the machine at which the application is running.
- A server directory cache that is allocated and used for some of the internal database manager processes.

For application directory caches, when an application issues its first connect, each directory file is read and the information is cached in private memory for this application. The cache is used by the application process on subsequent connect requests and is maintained for the life of the application process. If a database is not found in the application directory cache, the directory files are searched for the information, but the cache is not updated. If the application modifies a directory entry, the next connect within that application will cause the cache for this application to be refreshed. The application directory cache for other applications will not be refreshed. When the application process terminates, the cache is freed. (To refresh the directory cache used by a command line processor session, issue a **db2 terminate** command.)

For server directory caches, when a database manager instance is started (**db2start**), each directory file is read and the information is cached in the server memory. This cache is maintained until the instance is stopped (**db2stop**). If a directory entry is not found in this cache, the directory files are searched for the information. Normally, this server directory cache is never refreshed during the time the instance is running. However, an offline backup marks the server directory cache as invalid and will refresh the cache even with the instance running.

Recommendation: Use directory caching if your directory files do not change frequently and performance is critical.

In addition, on remote clients, directory caching can be beneficial if your applications issue several different connection requests. In this case, caching reduces the number of times a single application must read the directory files.

Directory caching can also improve the performance of taking database system monitor snapshots. In addition, you should explicitly reference the database name on the snapshot call, instead of using database aliases.

Note: Errors might occur when performing snapshot calls if directory caching is turned on and if databases are cataloged, uncataloged, created, or dropped after the database manager is started.

discover - Discovery mode

You can use this parameter to determine what kind of discovery requests, if any, the client can make.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

SEARCH [DISABLE, KNOWN, SEARCH]

From a client perspective, one of the following occurs:

- If you set **discover** to SEARCH, the client can issue search discovery requests to find Db2 server systems on the network. Search discovery provides a superset of the functionality provided by KNOWN discovery. If you set **discover** to SEARCH, both search and known discovery requests can be issued by the client.
- If you set **discover** to KNOWN, only known discovery requests can be issued from the client. By specifying some connection information for the administration server on a particular system, all the instance and database information on the Db2 system is returned to the client.
- If **discover** to DISABLE, discovery is disabled at the client.

discover_inst - Discover server instance

You can use this parameter to specify whether this instance can be detected by Db2 discovery.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

ENABLE [ENABLE, DISABLE]

If you set the **discover_inst** configuration parameter to ENABLE, the Db2 instance can be detected by Db2 discovery.

If you set the **discover_inst** configuration parameter to DISABLE, the Db2 instance cannot be detected by Db2 discovery.

fcm_buffer_size – Inter-member buffer size

This parameter specifies the size of the FCM (Fast Communications Manager) buffer.

The FCM buffer is used to send work units between members within a Db2 instance.

This parameter, measured in bytes, determines the size of the FCM buffer. The default size is 32KB.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients
- Db2 pureScale members

Parameter type

Configurable

Default [range]

32768 [4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576]

Updates to this parameter take effect only after you shut down and restart all members in an instance.

fcm_num_buffers - Number of FCM buffers

You can use this parameter to specify the number of 4KB buffers that are used for internal communications, referred to as messages, both among and within database servers.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server or Db2 pureScale database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

32-bit platforms

AUTOMATIC [895 - 65300]

64-bit platforms

AUTOMATIC [895 - 524288]

- Database server with local and remote clients: 1024
- Database with local clients: 895

- Partitioned database server or Db2 pureScale database server with local and remote clients: 4096

Fast communication manager (FCM) buffers are used for both inter-member and intra-member communications by default.

Important: The default value of the **fcm_num_buffers** parameter is subject to change by the Db2 Configuration Advisor after initial database creation.

You can set both an initial value and the AUTOMATIC value for the **fcm_num_buffers** configuration parameter. When you set the parameter to AUTOMATIC, FCM monitors resource usage and can increase or decrease resources if they are not used within 30 minutes. The amount that resources are increased or decreased depends on the operating system. On Linux operating systems, the number of buffers can be increased only 25% more than the starting value. If the database manager attempts to start an instance and cannot allocate the specified number of buffers, it decreases the number until it can start the instance.

If you want to set the **fcm_num_buffers** parameter to both a specific value and AUTOMATIC, and you do not want the system controller thread to adjust resources lower than the specified value, set the FCM_CFG_BASE_AS_FLOOR option of the **DB2_FCM_SETTINGS** registry variable to YES or TRUE. The **DB2_FCM_SETTINGS** registry variable value is adjusted dynamically.

If you are using multiple logical nodes, one pool of **fcm_num_buffers** buffers is shared by all the logical nodes on the same machine. You can determine the size of the pool by multiplying the value of the **fcm_num_buffers** parameter by the number of logical nodes on the physical machine. Examine the value that you are using; consider how many FCM buffers are allocated on a machine or machines with multiple logical nodes. If you have multiple logical nodes on the same machine, you might have to increase the value of the **fcm_num_buffers** parameter. The number of users on the system, the number of database partition servers on the system, or the complexity of the applications can cause a system to run out of message buffers.

fcm_num_channels - Number of FCM channels

This parameter specifies the number of FCM channels for each database partition.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server or Db2 pureScale database server with local and remote clients
- Satellite database server with local clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

UNIX 32-bit platforms

Automatic, with a starting value of 256, 512 or 2048 [128 - 120000]

UNIX 64-bit platforms

Automatic, with a starting value of 256, 512 or 2048 [128 - 524288]

Windows 32-bit

Automatic, with a starting value 10000 [128 - 120000]

Windows 64-bit

Automatic, with a starting value of 256, 512 or 2048 [128 - 524288]

The default starting values for different types of servers are as follows:

- For database server with local and remote clients, the starting value is 512.
- For database server with local clients, the starting value is 256.
- For partitioned database environment servers with local and remote clients, the starting value is 2048.

Fast communication manager (FCM) buffers are used for both inter-member and intra-member communications by default. To enable non-clustered database systems to use the FCM subsystem and the **fcm_num_channels** parameter, you had to set the **intra_parallel** parameter to YES

An FCM channel represents a logical communication end point between EDUs running in the Db2 engine. Both control flows (request and reply) and data flows (table queue data) rely on channels to transfer data between members.

When set to AUTOMATIC, FCM monitors channel usage, incrementally allocating and releasing resources as requirements change.

fcm_parallelism - Internode communication parallelism

This parameter specifies the degree of parallelism that is used for communication (both control messages and data flow) between members within a Db2 instance.

This parameter determines the number of sender and receiver fast communication manager conduit pairs. By default, an instance has only one pair: one sender and one receiver that handle all communication to and from other members in the instance.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

AUTOMATIC [AUTOMATIC, 1 - 8]

When this configuration parameter is set to 1, no parallelism is used. Updates to this parameter take effect only after you shut down and restart all members in an instance.

When this configuration parameter is set to **AUTOMATIC**, a computed value is determined based on the following items:

- Number of CPU cores
- Entries in the `db2nodes.cfg` file

fed_noauth - Bypass federated authentication

This parameter determines whether federated authentication will be bypassed at the instance.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

No [Yes; No]

When **fed_noauth** is set to yes, **authentication** is set to server or server_encrypt, and **federated** is set to yes, then authentication at the instance is bypassed. It is assumed that authentication will happen at the data source. Exercise caution when **fed_noauth** is set to yes. Authentication is done at neither the client nor at Db2. Any user who knows the SYSADM authentication name can assume SYSADM authority for the federated server.

federated - Federated database system support

This parameter enables or disables support for applications submitting distributed requests for data managed by data sources (such as the Db2 Family and Oracle).

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

No [Yes; No]

federated_async - Maximum asynchronous TQs per query configuration parameter

This parameter determines the maximum number of asynchrony table queues (ATQs) in the access plan that the federated server supports. The mechanism of ATQs does not work in Db2 pureScale environments.

Configuration type

Database manager

Applies to

- Partitioned database server with local and remote clients when federation is enabled.

Parameter type

Configurable online

Default [range]

0 [0 to 32 767 inclusive, -1, ANY]

When ANY or -1 is specified, the optimizer determines the number of ATQs for the access plan. The optimizer assigns an ATQ to all eligible SHIP or remote pushdown operators in the plan. The value that is specified for DB2_MAX_ASYNC_REQUESTS_PER_QUERY server option limits the number of asynchronous requests.

Recommendation

The **federated_async** configuration parameter supplies the default or starting value for the special register and the bind option. You can override the value of this parameter by setting the value of the CURRENT FEDERATED ASYNCHRONY special register, **FEDERATED_ASYNC** bind option, or **FEDERATED_ASYNC** precompile option to a higher or a lower number.

If the special register or the bind option do not override the **federated_async** configuration parameter, the value of the parameter determines the maximum number of ATQs in the access plan that the federated server allows. If the special register or the bind option overrides this parameter, the value of the special register or the bind option determines the maximum number of ATQs in the plan.

Any changes to the **federated_async** configuration parameter affect dynamic statements as soon as the current unit of work commits. Subsequent dynamic statements recognize the new value automatically. A restart of the federated database is not needed. Embedded SQL packages are not invalidated nor implicitly rebound when the value of the **federated_async** configuration parameter changes.

If you want the new value of the **federated_async** configuration parameter to affect static SQL statements, you need to rebind the package.

fenced_pool - Maximum number of fenced processes

This parameter represents the number of threads cached in each **db2fmp** process for threaded **db2fmp** processes (processes serving threadsafe stored procedures and UDFs). For non-threaded **db2fmp** processes, this parameter represents the number of processes cached.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Default [range]

-1 (**max_coordagents**), Automatic [-1; 0-64 000]

Unit of measure

Counter

Restrictions:

- If this parameter is updated dynamically, and the value is decreased, the database manager does not proactively terminate **db2fmp** threads or processes, instead it stops caching them as they are used in order to reduce the number of cached db2fmp's down to the new value.
- If this parameter is updated dynamically, and the value is increased, the database manager caches more **db2fmp** threads and processes when they are created.
- When this parameter is set to -1, the default, it assumes the value of the **max_coordagents** configuration parameter. Note that only the value of **max_coordagents** is assumed and not the automatic setting or behavior.
- When this parameter is set to AUTOMATIC, also the default:
 - The database manager allows the number of **db2fmp** threads and processes cached to increase based on the high water mark of coordinating agents. Specifically, the automatic behavior of this parameter allows it to grow depending on the maximum number of coordinating agents the database manager has ever registered, at the same time, since it started.
 - The value assigned to this parameter represents a lower bound for the number of **db2fmp** threads and process to cache.

Recommendation: If your environment uses fenced stored procedures or user defined functions, then this parameter can be used to ensure that an appropriate number of **db2fmp** processes are available to process the maximum number of concurrent stored procedures and UDFs that run on the instance, ensuring that no new fenced mode processes need to be created as part of stored procedure and UDF execution.

If you find that the default value is not appropriate for your environment because an inappropriate amount of system resource is being given to **db2fmp** processes and is affecting performance of the database manager, the following procedure might be useful in providing a starting point for tuning this parameter:

```
fenced_pool = # of applications allowed to make stored procedure and
UDF calls at one time
```

If **keepfenced** is set to YES, then each **db2fmp** process that is created in the cache pool will continue to exist and will use system resources even after the fenced routine call has been processed and returned to the agent.

If **keepfenced** is set to NO, then non-threaded db2fmp processes will terminate when they complete execution, and there is no cache pool. Multithreaded **db2fmp** processes will continue to exist, but no threads will be pooled in these processes. This means that even when **keepfenced** is set to NO, you can have one threaded C **db2fmp** process and one threaded Java™ **db2fmp** process on your system.

In previous versions, this parameter was known as **maxdari**.

group_plugin - Group plug-in

This parameter specifies the name of the group plug-in library.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid string]

By default, this value is null, and Db2 uses the operating system group lookup. The plug-in will be used for all group lookups. For non-root installations, if the Db2 userid and password plug-in library is used, the **db2rfe** command must be run before using your Db2 product.

health_mon - Health monitoring

This parameter allows you to specify whether you want to monitor an instance, its associated databases, and database objects according to various health indicators.

Important: This parameter is deprecated in Version 10.1 and might be removed in a future release. This parameter can still be used in releases before Version 10.1.

You cannot use this parameter in Db2 pureScale environments.

Configuration type

Database manager

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Off [On; Off]

Related Parameters

If **health_mon** is turned on, an agent will collect information about the health of the objects you have selected. If an object is considered to be in an unhealthy position, based on thresholds that you have set, notifications can be sent, and actions can be taken automatically. If **health_mon** is turned off, the health of objects will not be monitored.

You can use the CLP to select the instance and database objects that you want to monitor. You can also specify where notifications should be sent, and what actions should be taken, based on the data collected by the health monitor.

indexrec - Index re-creation time

This parameter indicates when the database manager attempts to rebuild invalid indexes, and whether or not any index build is redone during rollforward or high availability disaster recovery (HADR) log replay on the standby database.

Configuration type

Database and Database Manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]**UNIX Database Manager**

restart [restart; restart_no_redo; access; access_no_redo]

Windows Database Manager

restart [restart; restart_no_redo; access; access_no_redo]

Database

Use system setting [system; restart; restart_no_redo; access; access_no_redo]

There are five possible settings for this parameter:

SYSTEM

Use system setting specified in the database manager configuration file to decide when invalid indexes are rebuilt, and whether any index build log records are to be redone during rollforward or HADR log replay.

Note: This setting is only valid for database configurations.

ACCESS

Invalid indexes are rebuilt when the underlying table is first accessed. Any fully logged index builds are redone during rollforward or HADR log replay. When HADR is started and an HADR takeover occurs, any invalid indexes are rebuilt after takeover when the underlying table is first accessed.

ACCESS_NO_REDO

Invalid indexes are rebuilt when the underlying table is first accessed. Any fully logged index build is not redone during rollforward or HADR log replay and those indexes are left invalid. When HADR is started and an HADR takeover takes place, any invalid indexes are rebuilt after takeover when the underlying table is first accessed. Access to the underlying tables on the new primary cause index rebuild, which causes log records to be written and then sent to the new standby, which in turn causes the indexes to be invalidated on the standby.

RESTART

The default value for **indexrec**. Invalid indexes are rebuilt when a **RESTART DATABASE** command is either explicitly or implicitly issued. Any fully logged index build will be redone during rollforward or HADR log replay. When HADR is started and an HADR takeover takes place, any invalid indexes will be rebuilt at the end of takeover.

Note: In a Db2 pureScale environment, indexes are rebuilt only during a group crash recovery, not as part of member crash recovery.

Note: When a database terminates abnormally while applications are connected to it, and the `autorestart` parameter is enabled, a **RESTART DATABASE** command is implicitly issued when an application connects to a database. If the database terminates normally, then the **RESTART DATABASE** command must be issued before any connections are made to the database, otherwise index recreation does not take place. If the command is not issued, the invalid indexes are rebuilt the next time the underlying table is accessed.

RESTART_NO_REDO

Invalid indexes are rebuilt when a **RESTART DATABASE** command is either explicitly or implicitly issued. Any fully logged index build is not redone during rollforward or HADR log replay and instead those indexes are rebuilt when rollforward completes or when HADR takeover takes place. Takeover causes index rebuild on underlying tables on the new primary, which causes log records to be written and then sent to the new standby, which in turn causes the indexes to be invalidated on the standby.

When a database terminates abnormally while applications are connected to it, and the `autorestart` parameter is enabled, a **RESTART DATABASE** command is implicitly issued when an application connects to a database. If the database terminates normally, then the **RESTART DATABASE** command must be issued before any connections are made to the database, otherwise index recreation does not take place. If the command is not issued, the invalid indexes are rebuilt the next time the underlying table is accessed.

Indexes can become invalid when fatal disk problems occur. If this happens to the data itself, the data could be lost. However, if this happens to an index, the index can be recovered by recreating it. If an index is rebuilt while users are connected to the database, two problems could occur:

- An unexpected degradation in response time might occur as the index file is re-created. Users accessing the table and using this particular index would wait while the index was being rebuilt.
- Unexpected locks might be held after index recreation, especially if the user transaction that caused the index to be re-created never performed a `COMMIT` or `ROLLBACK`.

Recommendation: The best choice for this option on a high-user server and if restart time is not a concern, would be to have the index rebuilt at **DATABASE RESTART** time as part of the process of bringing the database back online after a crash.

Setting this parameter to `ACCESS` or to `ACCESS_NO_REDO` result in a degradation of the performance of the database manager while the index is being re-created. Any user accessing that specific index or table would have to wait until the index is re-created.

If this parameter is set to `RESTART`, the time taken to restart the database is longer due to index re-creation, but normal processing would not be impacted once the database has been brought back online.

The difference between the `RESTART` and the `RESTART_NO_REDO` values, or between the `ACCESS` and the `ACCESS_NO_REDO` values, is only significant when full logging is activated for index build operations, such as **CREATE INDEX** and **REORG INDEX** operations, or for an index rebuild. You can activate logging by enabling the **logindexbuild** database configuration parameter or by enabling `LOG INDEX BUILD`

when altering a table. By setting **indexrec** to either RESTART or ACCESS, operations involving a logged index build can be rolled forward without leaving the index object in an invalid state, which would require the index to be rebuilt at a later time.

instance_memory - Instance memory

This parameter specifies the maximum amount of memory that can be allocated for a database partition. If you are using Db2 database products with memory usage restrictions or if you set it to a specific value. Otherwise, the AUTOMATIC setting allows instance memory to grow as needed.

Configuration type

Database manager.

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online (requires an instance attachment).

Configurable by member in a Db2 pureScale environment and in partitioned database environments.

Default [range]

AUTOMATIC [0 - system memory capacity].

The Db2 license memory limit of the installed product further restricts the maximum value. 32 bit instances are also restricted to a maximum of 1,000,000.

Unit of measure

Pages (4 KB).

When allocated

Not applicable.

When freed

Not applicable.

The default value of the **instance_memory** parameter is AUTOMATIC. The AUTOMATIC setting results in a value that is computed at database partition activation. The computed value ranges between 75 percent and 95 percent of the system memory capacity on the system - the larger the system, the higher the percentage. For Db2 database products with memory usage restrictions, the computed value is also limited by the maximum that is allowed by the product license. For database partition servers with multiple logical database partitions, this computed value is divided by the number of logical database partitions.

Possible values for **instance_memory** are:

- AUTOMATIC- This value is the default value. The AUTOMATIC setting results in a value that is computed at database partition activation. The computed value ranges between 75 percent and 95 percent of the system memory capacity on the system - the larger the system, the higher the percentage. For Db2 database products with memory usage restrictions, the computed value is also limited by the maximum that is allowed by the product license. For database partition servers with multiple logical database partitions, this computed value is divided by the number of logical database partitions.

The AUTOMATIC setting allows Instance Memory usage to grow as needed. If STMM is enabled and tunes the overall database memory size, STMM tunes based on available system memory. This option indirectly determines the actual instance memory usage. The number of 4 KB pages is calculated, but applies only to a configuration, which has a license memory limit. In this case, the calculated `instance_memory` value is limited by the license memory limit.

- 0 - Use this value only to reset a member-specific setting back to the global setting.
- 1 - 100 - Specifies the `instance_memory` limit by calculating the percentage of available RAM divided by the number of local partitions. The in-memory value is updated at member startup time to reflect the calculated number of 4 KB pages. Used to set the Db2 percentage consumption of the total RAM on the workstation in Db2 instances with heterogeneous workstation hardware configurations.
- 101 - system memory capacity - Specifies the memory limit as the number of 4 KB pages. This option also represents a tuning target if STMM is enabled.

Updating the `instance_memory` parameter dynamically

- Dynamic updates to the `instance_memory` parameter require an instance attachment. See the **ATTACH** command for details.
- For Db2 database products with memory usage restrictions, dynamic updates to `instance_memory` must indicate a value less than any license limit or AUTOMATIC. Otherwise, the update fails and the SQL5130N error message is returned.
- Dynamic updates to the `instance_memory` parameter must indicate a value less than the amount of system memory capacity or AUTOMATIC. Otherwise, the update is deferred until the next **db2start** is issued and the SQL1362W warning message is returned.
- Dynamic updates to the `instance_memory` parameter must indicate a value larger than the current amount of in-use instance memory. Otherwise, the update is deferred until the instance is restarted, and the SQL1362W warning message is returned. The amount of in-use instance memory can be determined by subtracting the cached memory value from current usage value in the output of the **db2pd -dbptnmem** command. The minimum value would be the highest in-use instance memory across all database partitions.
- If the `instance_memory` parameter is set to a value greater than the amount of system memory capacity, the next **db2start** command that being issued fails, and return the SQL1220N error message.
- If the `instance_memory` parameter is dynamically updated to AUTOMATIC, the value is recalculated immediately.

Suggestions for using fixed instance memory limits with a multi-member instance

Specifying a member number allows a different limit to be set on the specified member. Otherwise, the global value applies to all members. Since each member might have different memory requirements, fixed instance memory limits must be set carefully for each member. The following factors might be considered:

- Only fixed limits can be set when you are updating a member.
- When fixed instance memory limits are set in Db2 pureScale environments, ensure that STMM is configured to run independently on each member.

- It is suggested that you do not use STMM in a partitioned database environment with fixed instance memory limits.
- When the member clause is used, the **instance_memory** parameter value cannot be set to AUTOMATIC at the same time.

You can update a member's instance memory setting back to a global value by specifying the following values:

```
update dbm cfg member n using instance_memory 0
```

Where n is the member number.

Controlling Db2 Memory consumption

Db2 memory consumption varies depending on workload and configuration. In addition to this factor, self-tuning of the **database_memory** becomes a factor if it is enabled. Self-tuning of the **database_memory** is enabled when **database_memory** is set to AUTOMATIC and the self-tuning memory manager (STMM) is active.

If the instance is running on a Db2 database product without memory usage restrictions and the **instance_memory** parameter is set to AUTOMATIC, an instance memory limit is not enforced. The database manager allocates system memory as needed. If the self-tuning of **database_memory** is enabled, STMM updates the configuration to achieve optimal performance while it monitors available system memory. The monitoring of available memory ensures that system memory is not over-committed.

If the instance is running on a Db2 database product with memory usage restrictions or **instance_memory** is set to a specific value, an instance memory limit is enforced. The database manager allocates system memory up to this limit. The application can receive memory allocation errors when this limit is reached. More considerations are as follows:

- If the self-tuning of **database_memory** is enabled and the **instance_memory** parameter is set to a specific value, STMM updates the configuration to achieve optimal performance while it maintains sufficient free instance memory. This behavior ensures that enough instance memory is available to satisfy volatile memory requirements. System memory is not monitored.
- If the self-tuning of **database_memory** is enabled and the **instance_memory** parameter is set to AUTOMATIC, a **instance_memory** parameter limit is enforced for Db2 database product with memory usage restrictions. STMM updates the configuration to achieve optimal performance while it maintains available system memory and maintaining sufficient free instance memory.

Monitoring Instance Memory usage

Use the **db2pd -dbptnmem** command to show details on instance memory usage.

Use the new ADMIN_GET_MEM_USAGE table function to get the total instance memory consumption by a Db2 instance for a specific database partition, or for all database partitions. This table function also returns the current upper bound value.

When Tivoli® Storage FlashCopy® Manager shared memory is allocated, each local database partition's share of the overall shared memory size for the system is accounted for in that database partition's **instance_memory** usage.

Usage notes

- The **instance_memory** configuration parameter can be configured for individual members by using the "for member" clause as part of the cfg update. If a value is specified for a member, that value applies only for that member. If no value is specified for a member, then the global value of **instance_memory** is used.

The CLP output that is returned by db2 get dbm cfg shows what is applicable on the current member. Whether the member is using the global value or a per member override, the output shows only the currently in use value.

If **instance_memory** is configured per-member on DPF or Db2 pureScale instances, then the deferred value for individual members is returned as the **member_inst_mem** configuration parameter name rather than through **instance_memory** for individual members.

Example

```
$ db2 "update dbm cfg member 1 using instance_memory 12345678 immediate"
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed successfully.
```

```
$ db2 "select name, current dbpartitionnum as member, varchar(VALUE, 20) VALUE, varchar(DEFERRED_VALUE, 20) DEFERRED_VALUE
from SYSIBMADM.DBMCFG where NAME = 'instance_memory' or NAME = 'member_inst_mem'"
```

```
NAME MEMBER VALUE DEFERRED_VALUE
```

```
-----
instance_memory 1 12345678 30914469
member_inst_mem 1 12345678 12345678
```

```
2 record(s) selected.
```

intra_parallel - Enable intrapartition parallelism

This parameter specifies whether or not database connections will use intrapartition query parallelism by default.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

NO (0) [SYSTEM (-1), NO (0), YES (1)]

A value of YES enables intrapartition query parallelism. A value of NO disables intrapartition query parallelism.

A value of SYSTEM causes the parameter value to be set to YES or NO based on the hardware on which the database manager is running. If the number of logical CPUs on the system is > 1, when the value is set to SYSTEM, intrapartition query parallelism is enabled.

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Note:

- Parallel index creation does not use this configuration parameter.

- If you change this parameter value, packages might be rebound to the database, and some performance degradation might occur.
- The **intra_parallel** setting can be overridden in an application by a call to the ADMIN_SET_INTRA_PARALLEL procedure. Both the **intra_parallel** setting and the value set in an application by the ADMIN_SET_INTRA_PARALLEL procedure can be overridden in a workload by setting the MAXIMUM DEGREE attribute in a workload definition.

java_heap_sz - Maximum Java interpreter heap size

You can use this parameter to determine the maximum size of the heap that is used by the Java interpreter started to service Java Db2 stored procedures and UDFs.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

HP-UX

4096 [0 - 524 288]

All other operating systems

2048 [0 - 524 288]

Unit of measure

Pages (4 KB)

When allocated

When a Java stored procedure or UDF starts

When freed

When the **db2fmp** process (fenced) or the **db2agent** process (trusted) terminates.

There is one heap per db2fmp process running a Java stored procedure. For multithreaded db2fmp processes, multiple applications using threadsafe fenced routines are serviced from a single heap. In all situations, only the processes that run Java UDFs or stored procedures ever allocate this memory. On partitioned database systems, the same value is used at each database partition.

XML data is materialized when passed to stored procedures as IN, OUT, or INOUT parameters. When you are using Java stored procedures, the heap size might need to be increased based on the quantity and size of XML arguments, and the number of external stored procedures that are being executed concurrently.

jdk_path - Software Developer's Kit for Java installation path

You can use this parameter to specify the directory under which the Software Developer's Kit (SDK) for Java is installed. The Java SDK is used for running Java

stored procedures and user-defined functions. The CLASSPATH and other environment variables used by the Java interpreter are computed from the value of this parameter.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

NULL [VALID_PATH]

If the SDK for Java was installed with your Db2 product, this parameter is set properly. However, if you reset the database manager configuration parameters by issuing the **dbm cfg** command, you need to specify where the SDK for Java is installed.

keepfenced - Keep fenced process

This parameter indicates if a fenced mode process is kept after a fenced mode routine call is complete. Fenced mode processes are created as separate system entities in order to isolate user-written fenced mode code from the database manager agent process. This parameter is only applicable on database servers.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

s

Parameter type

Configurable

Default [range]

Yes [Yes; No]

If **keepfenced** is set to No, and the routine being executed is not threadsafe, a new fenced mode process is created and destroyed for each fenced mode invocation. If **keepfenced** is set to no, and the routine being executed is threadsafe, the fenced mode process persists, but the thread created for the call is terminated. If **keepfenced** is set to yes, a fenced mode process or thread is reused for subsequent fenced mode calls. When the database manager is stopped, all outstanding fenced mode processes and threads will be terminated.

Setting this parameter to yes will result in additional system resources being consumed by the database manager for each fenced mode process that is activated, up to the value contained in the **fenced_pool** parameter. A new process is only created when no existing fenced mode process is available to process a subsequent fenced routine invocation. This parameter is ignored if **fenced_pool** is set to 0.

Recommendation: In an environment in which the number of fenced mode requests is large relative to the number of non-fenced mode requests, and system resources are not constrained, then this parameter can be set to `yes`. This will improve the fenced mode process performance by avoiding the initial fenced mode process creation processing time since an existing fenced mode process will be used to process the call. In particular, for Java routines, this will save the cost of starting the Java Virtual Machine (JVM), a very significant performance improvement.

For example, in an OLTP debit-credit banking transaction application, the code to perform each transaction could be performed in a stored procedure which executes in a fenced mode process. In this application, the main workload is performed out of fenced mode processes. If this parameter is set to `no`, each transaction incurs the additional processing time of creating a new fenced mode process, resulting in a significant performance reduction. If, however, this parameter is set to `yes`, each transaction would try to use an existing fenced mode process, which would avoid this additional processing time.

In previous versions of Db2, this parameter was known as `keepdari`.

encrypted_database - Database encryption state

This parameter indicates whether the database is encrypted. It is set when the database is created and cannot be changed.

Configuration type

Database

Parameter type

Informational

Default [range]

NO [NO, YES]

Valid values for this parameter are:

YES The database is encrypted.

NO The database is not encrypted. This is the default.

keystore_location - Keystore location

This parameter specifies the location of the keystore that is used to store encryption keys or remote storage account credentials.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

NULL [NULL, any valid file name]

Valid values for this parameter are:

NULL There is no keystore defined for this instance, and no databases under this instance are encrypted. You cannot set **keystore_location** to NULL unless the **keystore_type** database manager configuration parameter is set to NULL.

filename

- When the **keystore_type** database manager configuration parameter is set to PKCS12, this parameter specifies the absolute file name of the keystore file.
- When the **keystore_type** is set to KMIP, this parameter is the absolute filename of the centralized keystore configuration file.
- When the **keystore_type** is set to PKCS11, this parameter is the absolute filename of the PKCS #11 keystore configuration file.

keystore_type - Keystore type

This parameter specifies the type of keystore that is used to store encryption keys or remote storage account credentials.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

NONE [NONE, PKCS12, KMIP, PKCS11]

Valid values for this parameter are:

NONE

There is no keystore defined for this instance, and no databases under this instance are encrypted.

PKCS12

Specifies to use a local keystore provided by IBM® Global Security Kit (GSKit). The value of the **keystore_location** database manager configuration parameter is used to configure the location of the keystore.

You cannot set **keystore_type** to PKCS12 unless the **keystore_location** database manager configuration parameter is set to a non-NULL file name.

Key Management Interoperability Protocol (KMIP)

Specifies to use a centralized keystore provided by a key manager that supports the Key Management Interoperability Protocol (KMIP) 1.1. The **keystore_location** configuration parameter is used to configure the absolute path of a centralized keystore configuration file.

You cannot set **keystore_type** to KMIP unless **keystore_location** is set to an absolute path of a centralized keystore configuration file.

PKCS11

Specifies to use a PKCS #11 keystore provided by a key manager that supports PKCS #11. The **keystore_location** configuration parameter is used to configure the absolute path of a PKCS #11 keystore configuration file.

You cannot set **keystore_type** to PKCS11 unless **keystore_location** is set to an absolute path of a PKCS #11 keystore configuration file.

local_gssplugin - GSS API plug-in used for local instance level authorization

This parameter specifies the name of the default GSS API plug-in library to be used for instance level local authorization when the value of the **authentication** database manager configuration parameter is set to GSSPLUGIN or GSS_SERVER_ENCRYPT.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid string]

max_connections - Maximum number of client connections

This parameter indicates the maximum number of client connections allowed per member.

Configuration type

Database manager

Parameter type

Configurable online

Applies to

- Database server with local and remote clients
- Database server with local clients
- Database Server or Connect server with local and remote clients (for **max_connections**, **max_coordagents**, **num_initagents**, **num_poolagents**, and also **federated_async**, if you are using a Federated environment)

Default [range]

-1 and AUTOMATIC (**max_coordagents**) [-1 and AUTOMATIC, 1 - 64000]

A setting of -1 means that the value associated with **max_coordagents** will be used, not the automatic setting or behavior. AUTOMATIC means that the database manager picks the value for this parameter using whatever technique works best. AUTOMATIC is an ON/OFF switch in the configuration file and is independent of the value, hence both -1 and AUTOMATIC can be the default setting.

For details, see: “Restrictions and behavior when configuring max_coordagents and max_connections” on page 24.

The Concentrator

The Concentrator is OFF when **max_connections** is equal to or less than **max_coordagents**. The Concentrator is ON when **max_connections** is greater than **max_coordagents**.

This parameter controls the maximum number of client applications that can be connected to a member in the instance. Typically, each application is assigned a coordinator agent. The agent facilitates the operations between the application and the database. When the default value for this parameter is used, the Concentrator feature is not activated. As a result, each agent operates within its own private memory and shares database manager and database global resources, such as the buffer pool, with other agents. When the parameter is set to a value greater than the default, the Concentrator feature is activated.

Usage

The maximum number of client connections can be larger than the value set by the **max_connections** configuration parameter if the user holds one of the following authorities:

- SYSADM
- SYSCTRL
- SYSMAIN

Note: Bypassing the maximum number of client connections set by the **max_connections** configuration parameter is useful for many reasons including the following tasks:

- Administrative tasks
- Capturing snapshot information
- Critical troubleshooting tasks
- Maintenance tasks (such as forcing users off the system)

max_connretries - Node connection retries

This parameter specifies the maximum number of times an attempt will be made to establish a network connection between two Db2 members.

Configuration type

Database manager

Applies to

Partitioned database server with local and remote clients

Db2 pureScale server

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

5 [0-100]

If the attempt to establish communication between two Db2 members fails (for example, the value specified by the **conn_elapse** parameter is reached), **max_connretries** specifies the number of connection retries that can be made to a Db2 member. If the value specified for this parameter is exceeded, an error is returned.

max_coordagents - Maximum number of coordinating agents

You can use this parameter to limit the number of coordinating agents.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Default [range]

200, AUTOMATIC [-1, 1 - 64 000]

A setting of -1 translates into a value of 200.

For details, see: “Restrictions and behavior when configuring max_coordagents and max_connections” on page 24.

The Concentrator

When the Concentrator is OFF, that is, when **max_connections** is equal to or less than **max_coordagents**, this parameter determines the maximum number of coordinating agents that can exist at one time on a server node.

One coordinating agent is acquired for each local or remote application that connects to a database or attaches to an instance. Requests that require an instance attachment include **CREATE DATABASE**, **DROP DATABASE**, and Database System Monitor commands.

When the Concentrator is ON, that is, when **max_connections** is greater than **max_coordagents**, there might be more connections than coordinator agents to service them. An application is in an active state only if there is a coordinator agent servicing it. Otherwise, the application is in an inactive state. Requests from an active application are serviced by the database coordinator agent and subagents in SMP or MPP configurations. Requests from an inactive application are queued until a database coordinator agent is assigned to service the application, when the application becomes active. As a result, this parameter might be used to control the load on the system.

Usage

The maximum number of coordinating agents can be larger than the value set by the **max_coordagents** configuration parameter if the user holds one of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAIN

Note: Bypassing the maximum number of coordinating agents set by the **max_coordagents** configuration parameter is useful for many reasons including the following tasks:

- Administrative tasks
- Capturing snapshot information
- Critical troubleshooting tasks
- Maintenance tasks, such as forcing users off the system

max_querydegree - Maximum query degree of parallelism

This parameter specifies the maximum degree of inrapartition parallelism that is used for any SQL statement executing on this instance of the database manager. An SQL statement will not use more than this number of parallel operations within a database partition when the statement is executed.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Statement boundary

Default [range]

-1 (ANY) [ANY, 1 - 32 767] (ANY means system-determined)

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

The **intra_parallel** configuration parameter must be set to YES to enable the database partition to use inrapartition parallelism for SQL statements. The **intra_parallel** parameter is no longer required for parallel index creation.

The default value for this configuration parameter is -1. This value means that the system uses the degree of parallelism determined by the optimizer; otherwise, the user-specified value is used.

Note: The degree of parallelism for an SQL statement can be specified at statement compilation time using the CURRENT DEGREE special register or the **DEGREE** bind option.

The maximum query degree of parallelism for an active application can be modified using the **SET RUNTIME DEGREE** command. The actual runtime degree used is the lower of:

- **max_querydegree** configuration parameter
- Application runtime degree
- SQL statement compilation degree

This configuration parameter applies to queries only.

max_time_diff - Maximum time difference between members

This parameter specifies the maximum time difference that is permitted between members in a Db2 pureScale environment that are listed in the node configuration file.

Configuration type

Database manager

Applies to

Members with local and remote clients

Parameter type

Configurable

Default [range]

In Db2 pureScale environments

1 [1 - 1 440]

Outside of Db2 pureScale environments

60 [1 - 1 440]

Unit of measure

Minutes

Each member has its own system clock. The time difference between two or more member system clocks is checked periodically. If the time difference between the system clocks is more than the amount specified by the **max_time_diff** parameter, warnings are logged in the db2diag log files.

In a Db2 pureScale environment, to ensure that members do not drift out of sync with each other, a Network Time Protocol (NTP) setup is required and periodically verified on each member. If chronyd or ntpd is not detected, warnings are logged in the db2diag log files.

The SQL1473N error message is returned in partitioned database environments where the system clock is compared to the virtual time stamp (VTS) saved in the SQL06CTL.LFH log control file. If the time stamp in the .LFH log control file is less than the system time, the time in the database log is set to the VTS until the system clock matches the VTS.

Db2 database manager uses Coordinated Universal Time (UTC), so different time zones are not a consideration when you set the **max_time_diff** parameter. UTC is the same as Greenwich Mean Time.

maxagents - Maximum number of agents

This parameter has been deprecated since Version 9.5, but is still being used by pre-Version 9.5 data servers and clients. Any value specified for this configuration parameter will be ignored by the database manager in Db2 Version 9.5 or later releases.

Note: The following information applies only to pre-Version 9.5 data servers and clients.

This parameter indicates the maximum number of database manager agents, whether coordinator agents or subagents, available at any given time to accept application requests

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

200 [1 - 64 000]

400 [1 - 64 000] on partitioned database server with local and remote clients

Unit of measure

Counter

If you want to limit the number of coordinating agents, use the **max_coordagents** parameter.

This parameter can be useful in memory constrained environments to limit the total memory usage of the database manager, because each additional agent requires additional memory.

Recommendation: The value of **maxagents** should be at least the sum of the values for **maxappls** in each database allowed to be accessed concurrently. If the number of databases is greater than the **numdb** parameter, then the safest course is to use the product of **numdb** with the largest value for **maxappls**.

Each additional agent requires some additional processing resources that are allocated at the time the database manager is started.

If you are encountering memory errors when trying to connect to a database, try making the following configuration adjustments:

- In a non-partitioned database environment with no intra-query parallelism enabled, increase the value of the **maxagents** database configuration parameter.
- In a partitioned database environment or an environment where intra-query parallelism is enabled, increase the larger of **maxagents** or **max_coordagents**.

maxcagents - Maximum number of concurrent agents

This parameter has been deprecated since Version 9.5, but is still being used by pre-Version 9.5 data servers and clients. Any value specified for this configuration parameter will be ignored by the database manager in Db2 Version 9.5 or later releases.

Note: The following information applies only to pre-Version 9.5 data servers and clients.

This parameter is used to control the load on the system during periods of high simultaneous application activity by limiting the maximum number of database manager agents that can be concurrently executing a database manager transaction

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]-1 (**max_coordagents**) [-1; 1 - **max_coordagents**]**Unit of measure**

Counter

This parameter does not limit the number of applications that can have connections to a database. It only limits the number of database manager agents that can be processed concurrently by the database manager at any one time, thereby limiting the usage of system resources during times of peak processing. For example, you might have a system requiring a large number of connections but with a limited amount of memory to serve those connections. Adjusting this parameter can be useful in such an environment, where a period of high simultaneous activity could cause excessive operating system paging.

A value of -1 indicates that the limit is **max_coordagents**.

Recommendation: In most cases the default value for this parameter will be acceptable. In cases where the high concurrency of applications is causing problems, you can use benchmark testing to tune this parameter to optimize the performance of the database.

mon_heap_sz - Database system monitor heap size

This parameter determines the amount of the memory, in pages, to allocate for database system monitor data. Memory is allocated from the monitor heap when database monitoring activities are performed.

Monitoring activities include turning on monitor switches, resetting monitor data, activating an event monitor or sending monitor events to an active event monitor.

With Version 9.5, this database configuration parameter has a default value of **AUTOMATIC**, meaning that the monitor heap can increase as needed until the **instance_memory** limit is reached.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Default [range]**32-bit platforms**

Automatic [0 - 60 000]

64-bit platforms

Automatic [0 - 2 147 483 647]

Unit of measure

Pages (4 KB)

When allocated

When the database manager is started with the **db2start** command

When freed

When the database manager is stopped with the **db2stop** command

A value of zero prevents the database manager from collecting database system monitor data.

Recommendation: The amount of memory required for monitoring activity depends on the number of monitoring applications (applications taking snapshots or event monitors), which switches are set, and the level of database activity.

If the configured memory in this heap runs out and no more unreserved memory is available in the instance shared memory region, one of the following events will occur:

- When the first application connects to the database for which this event monitor is defined, an error message is written to the administration notification log.
- If an event monitor being started dynamically using the SET EVENT MONITOR statement fails, an error code is returned to your application.
- If a monitor command or API subroutine fails, an error code is returned to your application.
- If an application needs to send an event monitor record but cannot allocate the record out of monitor heap, the application might be blocked until a record is allocated.

nodetype - Instance node type

This parameter specifies the type of instance (the type of database manager) created by Db2 products installed on your machine.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Informational

The possible values that are returned by this parameter and the products associated with that node type are shown in the following list:

- **Database server with local and remote clients** - a Db2 server product, supporting local and remote Data Server Runtime Clients, and capable of accessing other remote database servers.
- **Client** - a Data Server Runtime Client capable of accessing remote database servers.

- **Database server with local clients** - a Db2 relational database management system, supporting local Data Server Runtime Clients and capable of accessing other, remote database servers.
- **Partitioned database server with local and remote clients** - a Db2 server product, supporting local and remote Data Server Runtime Clients, and capable of accessing other remote database servers, and capable of parallelism.

notifylevel - Notify level

This parameter specifies the type of administration notification messages that are written to the administration notification log.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

3 [0 - 4]

On Linux and UNIX operating systems, the administration notification log is a text file called *instance.nfy*. On Windows, all administration notification messages are written to the Event Log. The errors can be written by Db2, the Health Monitor, the Capture and Apply programs, and user applications.

Valid values for this parameter are:

- 0 - No administration notification messages captured. (This setting is not recommended.)
- 1 - Fatal or unrecoverable errors. Only fatal and unrecoverable errors are logged. To recover from some of these conditions, you might need assistance from Db2 service.
- 2 - Immediate action required. Conditions are logged that require immediate attention from the system administrator or the database administrator. If the condition is not resolved, it could lead to a fatal error. Notification of very significant, non-error activities (for example, recovery) might also be logged at this level. This level will capture Health Monitor alarms. Informational messages will be shown at this level.
- 3 - Important information, no immediate action required. Conditions are logged that are non-threatening and do not require immediate action but might indicate a non-optimal system. This level will capture Health Monitor alarms, Health Monitor warnings, and Health Monitor attentions.
- 4 - Informational messages.

Usage Notes

- The administration notification log includes messages having values up to and including the value of **notifylevel**. For example, setting **notifylevel** to 3 will cause the administration notification log to include messages applicable to levels 1, 2, and 3.
- For a user application to be able to write to the notification file or Windows Event Log, it must call the db2AdminMsgWrite API.
- You might want to increase the value of this parameter to gather additional problem determination data to help resolve a problem. Note that you must set **notifylevel** to a value of 2 or higher for the Health Monitor to send any notifications to the contacts defined in its configuration.
- In specific circumstances, to display high importance messages, Db2 will override the **notifylevel** setting

num_initagents - Initial number of agents in pool

You can use this parameter to determine the initial number of idle agents that are created in the agent pool when the **db2start** command is issued.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Default [range]

0 [0 - 64 000]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

The database manager always starts the **num_initagents** idle agents as part of the **db2start** command, except if the value of this parameter is greater than **num_poolagents** during start up and **num_poolagents** is not set to AUTOMATIC. In this case, the database manager only starts the **num_poolagents** idle agents since there is no reason to start more idle agents than can be pooled.

num_initfenced - Initial number of fenced processes

This parameter indicates the initial number of non-threaded, idle **db2fmp** processes that are created in the **db2fmp** pool at **START DBM** time.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Default [range]

0 [0-64 000]

Setting this parameter will reduce the initial startup time for running non-threadsafe C and Cobol routines. This parameter is ignored if **keepfenced** is not specified.

It is much more important to set **fenced_pool** to an appropriate size for your system than to start up a number of **db2fmp** processes at **START DBM** time.

In previous versions, this parameter was known as **num_initdaris**.

num_poolagents - Agent pool size

This parameter sets the maximum size of the idle agent pool.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Default

100, AUTOMATIC [-1, 0 - 64000]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

This configuration parameter is set to **AUTOMATIC** with a value of 100 as the default. A setting of -1 is still supported, and translates into a value of 100. When this parameter is set to **AUTOMATIC**, the database manager automatically manages the number of idle agents to pool. Typically, this means that once an agent completes its work, it is not terminated, but becomes idle for a period of time. Depending on the workload and type of agent, it might be terminated after a certain amount of time.

When using **AUTOMATIC**, you can still specify a value for the **num_poolagents** configuration parameter. Additional idle agents will always be pooled when the current number of pooled idle agents is less than or equal to the value that you specified.

Examples

num_poolagents is set to 100 and AUTOMATIC

As an agent becomes free, it is added to the idle agent pool, where at some point the database manager evaluates whether it should be terminated or not. At the time when the database manager considers terminating the agent, if the total number of idle agents pooled is greater than 100, this agent will be terminated. If there are less than 100 idle agents, the idle agent will remain awaiting work. Using the **AUTOMATIC** setting allows additional idle agents beyond 100 to be pooled, which might be useful during periods of heavier system activity when the frequency of work can fluctuate on a larger scale. For cases where there are likely to be less than 100 idle agents at any given time, agents are guaranteed to be pooled. Periods of light system activity can benefit from this by incurring a less start up cost for new work.

num_poolagents is configured dynamically

If the parameter value is increased to a value greater than the number of pooled agents, the effects are immediate. As new agents become idle, they are pooled. If the parameter value is decreased, the database manager does not immediately reduce the number of agents in the pool. Rather, the pool size remains as it is, and agents are terminated as they are used and become idle again-gradually reducing the number of agents in the pool to the new limit.

Recommendation

For most environments the default of 100 and AUTOMATIC will be sufficient. If you have a specific workload where you feel too many agents are being created and terminated, you can consider increasing the value of **num_poolagents** while leaving the parameter set to AUTOMATIC.

numdb - Maximum number of concurrently active databases including host and System i® databases

This parameter specifies the number of local databases that can be concurrently active, or the maximum number of different database aliases that can be cataloged on a Db2 Connect server.

This parameter is overloaded.

The effect of the **numdb** database manager configuration parameter depends on the environment in which the parameter is used.

- For a Db2 Connect environment, you can specify the maximum number of database aliases that can be cataloged on a Db2 Connect server using the **numdb** database manager configuration parameter.
- Outside of a Db2 Connect environment, you can specify the number of local databases that can be concurrently active using the **numdb** database manager configuration parameter.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with only local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

UNIX Db2 pureScale environment

32 [1 - 200]

UNIX Database server with local and remote clients

32 [1 - 256]

UNIX Database server with only local clients

8 [1 - 256]

Windows Database server with local and remote clients

32 [1 - 256]

Windows Database server with only local clients

3 [1 - 256]

Unit of measure

Counter

Usage notes

- It is important to remember when setting or updating this parameter that each database takes up storage, and each active database requires an additional new shared memory segment.
- Set **numdb** to the default value if the number of databases being run on the instance is less than the default.
- Use of the **numdb** configuration parameter must be coordinated with the **db2_database_cf_memory** and **cf_db_mem_sz** configuration parameters.
- Changing the **numdb** parameter can impact the total amount of memory allocated. As a result, frequent updates to this parameter are not recommended. Plan the configurations for all parameters that are related to memory allocation for a database or an application connected to that database, and change them all at the same time.
- In addition to the **numdb** limit in a Db2 pureScale environment, there is a maximum number of database activations across all members in an instance. This maximum number is 512 database activations. As an example, if each member in a four member Db2 pureScale environment activated 200 databases, that is a total of 800 database member activations. Since 800 database activations exceeds the maximum upper limit, an error is returned.
- In a multiple database environment, if member crash recovery (MCR) is required, the number of databases that will be recovered in parallel on each member is set by the value of the **numdb** configuration parameter or the **DB2_MCR_RECOVERY_PARALLELISM_CAP** registry variable, whichever value is smaller.

query_heap_sz - Query heap size

This parameter specifies the maximum amount of memory that can be allocated for the query heap, ensuring that an application does not consume unnecessarily large amounts of virtual memory within an agent.

Important: This parameter is deprecated in Version 9.5 and might be removed in a future release. This parameter can still be used in pre-Version 9.5 data servers and clients. In Version 9.5 and later releases, the value specified for this configuration parameter is ignored.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

1 000 [2 - 524 288]

Unit of measure

Pages (4 KB)

When allocated

When an application (either local or remote) connects to the database

When freed

When the application disconnects from the database, or detaches from the instance

A query heap is used to store each query in the agent's private memory. The information for each query consists of the input and output SQLDA, the statement text, the SQLCA, the package name, creator, section number, and consistency token.

The query heap is also used for the memory allocated for blocking cursors. This memory consists of a cursor control block and a fully resolved output SQLDA.

The initial query heap allocated will be the same size as the application support layer heap, as specified by the **as1heapsz** parameter. The query heap size must be greater than or equal to two (2), and must be greater than or equal to the **as1heapsz** parameter. If this query heap is not large enough to handle a given request, it will be reallocated to the size required by the request (not exceeding **query_heap_sz**). If this new query heap is more than 1.5 times larger than **as1heapsz**, the query heap will be reallocated to the size of **as1heapsz** when the query ends.

Recommendation: In most cases the default value will be sufficient. As a minimum, you should set **query_heap_sz** to a value at least five times larger than **as1heapsz**. This will allow for queries larger than **as1heapsz** and provide additional memory for three or four blocking cursors to be open at a given time.

If you have very large LOBs, you might need to increase the value of this parameter so the query heap will be large enough to accommodate those LOBs.

release - Configuration file release level

This parameter specifies the release level of the configuration file.

Configuration type

Database manager, Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Informational

rstrt_light_mem - Restart light memory configuration parameter

This parameter specifies the maximum amount of memory that is allocated and reserved on a host for restart light recovery purposes. The amount is a percentage of **instance_memory** configuration parameter.

Configuration type

Database manager

Applies to

- Database server with local and remote clients

- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

AUTOMATIC [1 - 10 for AUTOMATIC; the user-defined range is 1 - 50]

Units of measure

Percentage of instance memory

When allocated

When the instance is started

When freed

When the instance is stopped

Having **rstrt_light_mem** set to AUTOMATIC means that when the instance is started, the Db2 database manager automatically calculates a fixed upper bound for the amount of memory to be pre-allocated and reserved for restart light recovery purposes. The specified amount of memory is reserved to accommodate failed members that need to be restarted on a host other than their home host (restart light mode). The Db2 database manager calculates the appropriate value based on the current setting for **instance_memory** and **numdb** (maximum number of currently active databases) as well as the number of members on the host . The automatically calculated value ranges between 1 and 10 percent of the instance memory limit and is included in the total amount of instance memory. Users can also explicitly set **rstrt_light_mem** to a value between 1 and 50 percent of the instance memory limit. Additional memory could result in an improvement in the recovery time, especially when there are multiple databases that need to be recovered. The additional memory will also result in less memory being available for normal work, reducing the throughput the members can handle.

The parameter is configurable but not online. Once the reserved recovery memory has been allocated when the Db2 instance is started, the amount of memory is fixed and will not change unless the configuration value is changed and the Db2 instance is globally stopped and started again.

To display information about the total amount of memory allocated on a host, use **db2pd** with the new **-totalmem** option. This information will also include the amount of restart light memory allocated on a host. Only information about the current host being accessed is returned, but **db2pd** can be run on separate hosts in parallel.

The **rstrt_light_mem** is only applicable to SD mode and does not display in ESE mode.

resync_interval - Transaction resync interval

This parameter specifies the time interval in seconds for which a transaction manager (TM), resource manager (RM) or sync point manager (SPM) should retry the recovery of any outstanding indoubt transactions found in the TM, the RM, or the SPM.

This parameter is applicable when you have transactions running in a distributed unit of work (DUOW) environment. This parameter also applies to recovery of federated database systems.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

180 [1 - 60 000]

Unit of measure

Seconds

Recommendation: If, in your environment, indoubt transactions will not interfere with other transactions against your database, you might want to increase the value of this parameter. If you are using a Db2 Connect gateway to access DRDA2 application servers, you should consider the effect indoubt transactions might have at the application servers even though there will be no interference with local data access. If there are no indoubt transactions, the performance impact will be minimal.

rqrioblk - Client I/O block size

This parameter specifies the block size at the Data Server Runtime Client when a blocking cursor is opened.

Configuration type

Database manager

Applies to

- Database server with remote clients
- Client
- Partitioned database server with remote clients

Parameter type

Configurable

Default [range]

65 535 [4 096 - 65 535]

Unit of measure

Bytes

The memory for blocked cursors is allocated out of the application's private address space, so you should determine the optimal amount of private memory to allocate for each application program. If the Data Server Runtime Client cannot allocate space for a blocking cursor out of an application's private memory, a non-blocking cursor will be opened.

You should also consider the effect of this parameter on the number and potential size of blocking cursors. Large row blocks might yield better performance if the number or size of rows being transferred is large (for example, if the amount of data is greater than 4 096 bytes). However, there is a trade-off in that larger record blocks increase the size of the working set memory for each connection.

Larger record blocks might also cause more fetch requests than are actually required by the application. You can control the number of fetch requests using the OPTIMIZE FOR clause on the SELECT statement in your application.

sheapthres - Sort heap threshold

This parameter represents a threshold on the total amount of private sort memory reservation available to sort-heap based operations on a member. Any sort memory reservation requests above this threshold might be reduced.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Switching between a 0 and non-0 setting is not an online change and requires restarting the instance. All other changes can be done online and requires an instance attachment.

Propagation class

Immediate

Default [Range]

32-bit platforms

0 [0, 250 - 2097152]

64-bit platforms

0 [0, 250 - 2147483647]

Unit of measure

Pages (4 KB)

While the **sortheap** parameter setting guides the maximum memory usage per operation, the **sheapthres** setting guides the overall private memory available to sort memory consumers per member. The **sheapthres** parameter guides the overall memory that is available by reducing the amount of sort reservation that is allowed when the total reservation requested by all activity on the member exceeds the **sheapthres** setting. The sort reservation that is granted is $(\text{sheapthres} / \text{total private sort requested} * \text{sortheap requested})$. At least 25% of the **sortheap** setting is always available to a memory consumer. The reservation request is not reduced below this amount.

There are three sort memory models possible. The model in use depends upon a number of elements in the configuration.

Shared sort memory model

The shared sort memory model is the default model and is in effect whenever **sheapthres** = 0. The **sheapthres** setting guides throttling for the private sort model, and a setting of 0 disables private sort memory. Under the shared sort model, all **sortheap** allocations are from the shared sort heap (**sheapthres_shr**), which is part of database shared memory (**database_memory**). The shared sort memory model is the only model where STMM tuning of **sortheap** and **sheapthres_shr** can occur.

Private sort memory model

The private sort memory model is active whenever **sheapthres** is not equal to zero and the configuration also does not enable shared sort memory. Under the private sort memory model, **sortheap** allocations are only allocated from private memory. Operations specifically requiring shared sort memory are not valid and return errors. No STMM sort-tuning takes place under this model.

Hybrid sort memory model

The hybrid sort memory model is active whenever **sheapthres** is not equal to zero, but the configuration dictates that shared sort memory is made available for certain operations. Operations not requiring shared sort memory are allocated from private memory. No STMM sort-tuning takes place under this model.

Any one of the following configuration settings can enable shared sort memory:

- Intra-parallelism is enabled (INTRA_PARALLEL = YES)
- Connection Concentrator is enabled (MAX_CONNECTIONS > MAX_COORDAGENTS)
- DB2_WORKLOAD=ANALYTICS

If shared sort memory is not enabled, the shared sort memory model and the hybrid sort memory models are not active and the following operations fail:

- Loading data into an XML table
- Column-organized query processing
- Applications requesting intra-parallel processing

If any of the operations in the preceding list fails, enable the shared sort memory model by completing all of the following steps:

- Set the **sheapthres** configuration parameter to 0.
- Recycle the Db2 instance. To recycle the Db2 instance, run the **db2stop** command followed by the **db2start** command.
- Configure the **sheapthres_shr** parameter appropriately.

Monitoring

There are numerous monitoring elements available.

For general monitoring, you can use the following monitoring elements:

- ACTIVE_SORTS
- TOTAL_SECTION_SORT_TIME
- TOTAL_SECTION_SORT_PROC_TIME
- TOTAL_SECTION_SORTS
- TOTAL_SORTS

For monitoring of constrained total sort memory configuration, you can use the following monitoring elements:

- POST_THRESHOLD_SORTS (private sort)
- POST_THRESHOLD_PEDS
- POST_THRESHOLD_PEAS
- POST_SHRTHRESHOLD_HASH_JOINS (private hash joins)

- POST_THRESHOLD_HASH_GRPBYs
- POST_THRESHOLD_OLAP_FUNCS
- SORT_OVERFLOWs
- TQ_SORT_HEAP_REJECTIONS
- SORT_HEAP_ALLOCATED (private sort reservations)

The SORT_HEAP_ALLOCATED monitoring element reflects reservation requests, not the actual amount of memory that is allocated. It is normal for operations to not fully allocate all of the requested reservation amounts.

For monitoring of private sort reservation levels, use the MON_GET_DATABASE routine. The following example shows a query that can be used to monitor the private sort reservation levels:

```
select SORT_HEAP_ALLOCATED from table (MON_GET_DATABASE (null))
```

This query returns the private sort memory reservation levels for the database in 4K units:

```
SORT_HEAP_ALLOCATED
-----
                128411
```

For monitoring private sort memory usage, use the MON_GET_MEMORY_POOL routine. The following example shows a query that can be used to monitor the private sort memory usage:

```
select edu_id, memory_pool_used, memory_pool_used_hwm
from table (mon_get_memory_pool(null,null,null))
where memory_pool_type='SORT'
```

This query returns the memory allocation levels for private sort pools in 1K units:

```
EDU_ID          MEMORY_POOL_USED    MEMORY_POOL_USED_HWM
-----
                2058             128             128
                2058             192             192
                2058            1280            1280
```

3 record(s) selected.

spm_log_file_sz - Sync point manager log file size

This parameter identifies the sync point manager (SPM) log file size in 4 KB pages.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

256 [4 - 1000]

Unit of measure

Pages (4 KB)

The log file is contained in the `spmlog` subdirectory under `sql11ib` and is created the first time SPM is started.

Recommendation: The sync point manager log file size should be large enough to maintain performance, but small enough to prevent wasted space. The size required depends on the number of transactions using protected conversations, and how often COMMIT or ROLLBACK is issued.

To change the size of the SPM log file:

1. Determine that there are no indoubt transactions by using the **LIST DRDA INDOUBT TRANSACTIONS** command.
2. If there are none, stop the database manager.
3. Update the database manager configuration with a new SPM log file size.
4. Go to the `$HOME/sql11ib` directory and issue `rm -fr spmlog` to delete the current SPM log. (Note: This shows the AIX command. Other systems might require a different remove or delete command.)
5. Start the database manager. A new SPM log of the specified size is created during the startup of the database manager.

spm_log_path - Sync point manager log file path

This parameter specifies the directory where the sync point manager (SPM) logs are written.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

NULL [any valid path or device]

If this parameter is null, by default, the logs are written to the `sql11ib/spmlog` directory, which, in a high-volume transaction environment, can cause an I/O bottleneck. Use this parameter to have the SPM log files placed on a faster disk than the current `sql11ib/spmlog` directory. This allows for better concurrency among the SPM agents.

Note: If SPM is enabled then the default directory will be created if it does not yet exist. To enable SPM, the configuration parameter **spm_name** must be set.

spm_max_resync - Sync point manager resync agent limit

This parameter identifies the number of agents that can simultaneously perform resync operations.

Configuration type

Database manager

Applies to

- Database server with local and remote clients

- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
20 [10 - 256]

spm_name - Sync point manager name

This parameter identifies the name of the sync point manager (SPM) instance to the database manager.

Configuration type
Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default
Derived from the TCP/IP hostname

srvcon_auth - Authentication type for incoming connections at the server

This parameter specifies how and where user authentication is to take place when handling incoming connections at the server; it is used to override the current authentication type.

Configuration type
Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
NOT_SPECIFIED [NOT_SPECIFIED; CLIENT; SERVER; SERVER_ENCRYPT;
DATA_ENCRYPT; DATA_ENCRYPT_CMP; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN;
GSS_SERVER_ENCRYPT]

If a value is not specified, Db2 uses the value of the **authentication** database manager configuration parameter.

For a description of each authentication type, see “authentication - Authentication type” on page 35.

srvcon_gssplugin_list - List of GSS API plug-ins for incoming connections at the server

This parameter specifies the GSS API plug-in libraries that are supported by the database server.

It handles incoming connections at the server when the **srvcon_auth** configuration parameter is specified as KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN or GSS_SERVER_ENCRYPT, or when **srvcon_auth** is not specified, and **authentication** is specified as KERBEROS, KRB_SERVER_ENCRYPT, GSSPLUGIN or GSS_SERVER_ENCRYPT.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid string]

By default, the value is null. If the authentication type is GSSPLUGIN and this parameter is NULL, an error is returned. If the authentication type is KERBEROS and this parameter is NULL, the Db2 supplied kerberos module or library is used. This parameter is not used if another authentication type is used.

When the authentication type is KERBEROS and the value of this parameter is not NULL, the list must contain exactly one Kerberos plug-in, and that plug-in is used for authentication (all other GSS plug-ins in the list are ignored). If there is more than one Kerberos plug-in, an error is returned.

Each GSS API plug-in name must be separated by a comma (,) with no space either before or after the comma. Plug-in names should be listed in the order of preference.

srvcon_pw_plugin - Userid-password plug-in for incoming connections at the server

This parameter specifies the name of the default userid-password plug-in library to be used for server-side authentication.

It handles incoming connections at the server when the **srvcon_auth** parameter is specified as CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT or DATA_ENCRYPT_CMP, or when **srvcon_auth** is not specified, and **authentication** is specified as CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT or DATA_ENCRYPT_CMP.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
Null [any valid string]

By default, the value is null and the userid-password plug-in library that is supplied with Db2 database is used. For non-root installations, if the Db2 userid and password plug-in library is used, the **db2rfe** command must be run before using your Db2 database product.

srv_plugin_mode - Server plug-in mode

This parameter specifies whether plug-ins are to run in fenced mode or unfenced mode. Unfenced mode is the only supported mode.

Configuration type
Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
UNFENCED

ssl_cipherspecs - Supported cipher specifications at the server configuration parameter

This configuration parameter specifies the cipher suites that the server allows for incoming connection requests when using SSL protocol.

Configuration type
Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
Null [TLS_RSA_WITH_AES_256_CBC_SHA; TLS_RSA_WITH_AES_128_CBC_SHA;
TLS_RSA_WITH_3DES_EDE_CBC_SHA; TLS_RSA_WITH_AES_256_GCM_SHA384;
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384;
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384;
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384;
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384;
TLS_RSA_WITH_AES_256_CBC_SHA256; TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA;
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA;
TLS_RSA_WITH_AES_128_GCM_SHA256; TLS_RSA_WITH_AES_128_CBC_SHA256;
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256;
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256;
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256;


```
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256;  
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA;  
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA;  
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA;  
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA]
```

You can specify multiple cipher specifications, such as TLS_RSA_WITH_AES_256_CBC_SHA or TLS_RSA_WITH_AES_128_CBC_SHA or TLS_RSA_WITH_3DES_EDE_CBC_SHA. They must be separated by a comma (,) with no space either before or after the comma.

During SSL handshake, if null or multiple values are specified, the client and the server negotiate and find the most secure cipher suites to use. If no compatible cipher suites is found, the connection fails. You cannot prioritize the cipher suites by specifying one before the another.

If you set **ssl_versions** to TLSv12, the following values are valid for **ssl_cipherspecs**.

- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA

ssl_clnt_keydb - SSL key file path for outbound SSL connections at the client configuration parameter

This configuration parameter specifies the key file to be used for SSL connection at the client-side.

Configuration type

Database manager

Applies to

- Database server with local and remote clients

- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid path; GSK_MS_CERTIFICATE_STORE]

This parameter specifies a fully qualified file path for the key file; or, on Windows only, the keyword `GSK_MS_CERTIFICATE_STORE` to use the Microsoft Windows Certificate Store.

The SSL key file has extension `.kbd` by default, and stores the signer certificate from the servers personal certificate. For a self-signed server personal certificate, the signer certificate is the public key. For a certificate authority signed server personal certificate, the signer certificate is the root CA certificate. The key file is accessed by the client to verify the servers personal certificate during the SSL handshake.

By default, the value is null. Depending on your application type, you should specify the client SSL key file path by the database manager configuration parameter `ssl_clnt_keydb`, the connection string `ssl_clnt_keydb`, or the `db2cli.ini` and `db2dsdriver.cfg` keyword `SSLClientKeystoredb` for a SSL connection request. If none of them is specified, the SSL connection fails.

ssl_clnt_stash - SSL stash file path for outbound SSL connections at the client configuration parameter

This configuration parameter specifies the fully qualified file path of the stash file to be used for SSL connections at the client-side.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid path]

The SSL stash file has extension `.sth` by default, and stores an encrypted version of the key database password. The password held in the stash file is used to access the SSL key file during an SSL connection request.

On Windows platforms, `ssl_clnt_stash` is not required if `ssl_clnt_keydb` is set to the keyword `GSK_MS_CERTIFICATE_STORE`.

By default the value is null. Depending on your application type, you can specify the client SSL stash file path by the database manager configuration parameter

`ssl_clnt_stash`, the connection string `ssl_clnt_stash`, or the `db2cli.ini` keyword `ssl_clnt_stash` for a SSL connection request. If none of them is specified, the SSL connection fails.

ssl_svr_keydb - SSL key file path for incoming SSL connections at the server configuration parameter

This configuration parameter specifies the key file to be used for SSL setup at server-side.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid path; GSK_MS_CERTIFICATE_STORE]

This parameter specifies a fully qualified file path of the key file or on Windows only, the keyword `GSK_MS_CERTIFICATE_STORE` which indicates to use Microsoft Windows Certificate Store.

The SSL key file has extension `.kdb` by default, and stores personal certificates, personal certificate requests and signer certificates. This key file is accessed during the instance startup and the servers personal certificate is sent to the client for server authentication during SSL handshake.

By default, the value is null. During the instance start up, you must define if the `DB2COMM` registry variable contains SSL. Otherwise, the instance starts up without SSL protocol support.

ssl_svr_label - Label in the key file for incoming SSL connections at the server configuration parameter

This configuration parameter specifies a label of the personal certificate of the server in the key database.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

Null

By default, the value is null. When establishing a SSL connection, the server certificate specified by this configuration parameter is sent to the client for server

authentication. If the value is null, the default certificate defined in the key file is used. If the default does not exist, the connection fails.

ssl_svr_stash - SSL stash file path for incoming SSL connections at the server configuration parameter

This configuration parameter specifies a fully qualified file path of the stash file to be used for SSL setup at server-side.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [any valid path]

The SSL stash file has extension `.sth` by default, and stores an encrypted version of the key database password. The password held in the stash file is used to access the SSL key file during the instance startup.

By default, the value is null. During the instance start up, you must define if the **DB2COMM** registry variable contains SSL. Otherwise, the instance starts up without SSL protocol support.

On Windows platforms, **ssl_svr_stash** is not required if **ssl_svr_keydb** is set to the keyword `GSK_MS_CERTIFICATE_STORE`.

start_stop_time - Start and stop timeout

This parameter specifies the time, in minutes, within which all database partition servers must respond to a **START DBM** or a **STOP DBM** command. It is also used as the timeout value during **ADD DBPARTITIONNUM** and **DROP DBPARTITIONNUM** operations.

Configuration type

Database manager

Applies to

Database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

10 [1 - 1 440]

Unit of measure

Minutes

Member or nodes that do not respond to **db2start** or **db2stop** commands within the specified time will be killed and cleaned up automatically by **db2start** or **db2stop** in a multi member/node instance. The diagnostic messages are logged

into the **diagpath** defined in the database manager configuration or at its default value (for example, `sqllib/db2dump/ $m` on UNIX operating systems).

If a **db2start** or **db2stop** operation in a multi-partition database is not completed within the value specified by the **start_stop_time** database manager configuration parameter, the database partitions that have timed out will be killed and cleaned up automatically. Environments with many database partitions with a low value for **start_stop_time** might experience this behavior. To resolve this behavior, increase the value of **start_stop_time**.

When adding a new database partition using one of the **db2start**, **START DATABASE MANAGER**, or **ADD DBPARTITIONNUM** commands, the add database partition operation must determine whether or not each database in the instance is enabled for automatic storage. This is done by communicating with the catalog partition for each database. If automatic storage is enabled, the storage path definitions are retrieved as part of that communication. Likewise, if system temporary table spaces are to be created with the database partitions, the operation might have to communicate with another database partition server to retrieve the table space definitions for the database partitions that reside on that server. These factors should be considered when determining the value of the **start_stop_time** parameter.

ssl_svcename - SSL service name configuration parameter

This configuration parameter specifies the name of the port that a database server uses to await communications from remote client nodes using SSL protocol.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

Null

This configuration parameter contains the port that a database server uses to await communications from remote client nodes through SSL protocol. This service name must be reserved for use by the database manager. During instance startup, you must define if the **DB2COMM** registry variable contains SSL. Otherwise the instance starts up without SSL protocol support.

If **DB2COMM** contains both TCPIP and SSL, the port specified by **ssl_svcename** must not be the same as the **svcename**. Otherwise, the instance starts up without either SSL or TCP/IP protocol support.

On UNIX operating systems, the services file is located in: `/etc/services`

The database server SSL port (number *n*) and its service name needs to be defined in the services file on the database client.

ssl_versions - Supported SSL versions at the server configuration parameter

This configuration parameter specifies Secure Sockets Layer (SSL) and Transport Layer Security (TLS) versions that the server supports for incoming connection requests.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

Null [TLSv1,TLSv12]

If you set the parameter to null or TLSv1, the parameter enables support for TLS version 1.0 (RFC2246) and TLS version 1.1 (RFC4346).

Note: During SSL handshake, the client and the server negotiate and find the most secure version to use either TLS version 1.0 or TLS version 1.1. If there is no compatible version between the client and the server, the connection fails. If the client supports TLS version 1.0 and TLS version 1.1, but the server support TLS version 1.0 only, then TLS version 1.0 is used.

If you set the parameter to TLSv12 (RFC5246), the parameter enables support for TLS version 1.2. This setting is required to comply with NIST SP 800-131A.

If you set the parameter to TLSv12 and TLSv1, the parameter enables support for TLS version 1.2 with the option to fall back on TLS version 1.0 and 1.1.

svcname - TCP/IP service name

This parameter contains the name of the TCP/IP port which a database server will use to await communications from remote client nodes. This name must be the reserved for use by the database manager.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

Null

In order to accept connection requests from a Data Server Runtime Client using TCP/IP, the database server must be listening on a port designated to that server.

The system administrator for the database server must reserve a port (number *n*) and define its associated TCP/IP service name in the services file at the server.

The database server port (number *n*) and its TCP/IP service name need to be defined in the services file on the database client.

On Linux and UNIX systems, the services file is located in: `/etc/services`

The **svcname** parameter should be set to the port number or the service name associated with the main connection port so that when the database server is started, it can determine on which port to listen for incoming connection requests.

If **DB2COMM** contains both TCP/IP and SSL, the port specified by **svcname** must not be the same as **ssl_svcname**. Otherwise, the instance starts up without either TCP/IP or SSL protocol support.

sysadm_group - System administration authority group name

This parameter defines the group name with SYSADM authority for the database manager instance.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

NULL

The SYSADM authority level is the highest level of administrative authority at the instance level. Users with SYSADM authority can run some utilities and issue some database and database manager commands within the instance.

SYSADM authority is determined by the security facilities used in a specific operating environment.

- For the Windows operating system, this parameter can be set to local or domain group. Group names must adhere to the length limits specified in SQL and XML limits. The following users have SYSADM authority if "NULL" is specified for **sysadm_group** database manager configuration parameter:
 - Members of the local Administrators group
 - Members of the Administrators group at the Domain Controller if **DB2_GRP_LOOKUP** is not set or set to DOMAIN
 - Members of DB2ADMNS group if Extended Security feature is enabled. The location of the DB2ADMNS group was decided during installation
 - The LocalSystem account
- For Linux and UNIX systems, if NULL is specified as the value of this parameter, the SYSADM group defaults to the primary group of the instance owner. If the value is not NULL, the SYSADM group can be any valid UNIX group name.

To restore the parameter to its default (NULL) value, use **UPDATE DBM CFG USING SYSADM_GROUP NULL**. You must specify the keyword NULL in uppercase.

sysctrl_group - System control authority group name

This parameter defines the group name with system control (SYSCTRL) authority. SYSCTRL has privileges allowing operations affecting system resources, but does not allow direct access to data.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

Null

Group names on all platforms are accepted as long as they adhere to the length limits specified in SQL and XML limits.

sysmaint_group - System maintenance authority group name

This parameter defines the group name with system maintenance (SYSMAINT) authority.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

Null

SYSMAINT has privileges to perform maintenance operations on all databases associated with an instance without having direct access to data.

Group names on all platforms are accepted as long as they adhere to the length limits specified in SQL and XML limits.

To restore the parameter to its default (NULL) value, use **UPDATE DBM CFG USING SYSMAINT_GROUP NULL**. You must specify the keyword NULL in uppercase.

sysmon_group - System monitor authority group name

This parameter defines the group name with system monitor (SYSMON) authority.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

Null

Users having SYSMON authority at the instance level have the ability to take database system monitor snapshots of a database manager instance or its databases. Refer to System monitor authority (SYSMON) for a full list of commands supported by SYSMON authority.

Users with SYSADM, SYSCTRL, or SYSMOINT authority automatically have the ability to take database system monitor snapshots and to use these commands.

Group names on all platforms are accepted as long as they adhere to the length limits specified in “SQL and XML limits” in the *Database Administration Concepts and Configuration Reference*.

To restore the parameter to its default (NULL) value, use **UPDATE DBM CFG USING SYSMON_GROUP NULL**. You must specify the keyword NULL in uppercase.

tm_database - Transaction manager database name

This parameter identifies the name of the transaction manager (TM) database for each Db2 instance.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

1ST_CONN [any valid database name]

A TM database can be:

- A local Db2 database
- A remote Db2 database that does not reside on a host or Power® System Server
- A Db2 for z/OS® Version 5 database if accessed via TCP/IP and the sync point manager (SPM) is not used.

The TM database is a database that is used as a logger and coordinator, and is used to perform recovery for indoubt transactions.

You can set this parameter to **1ST_CONN**, which will set the TM database to be the first database to which a user connects.

Recommendation: For simplified administration and operation, you might want to create a few databases over a number of instances and use these databases exclusively as TM databases.

tp_mon_name - Transaction processor monitor name

This parameter identifies the name of the transaction processing (TP) monitor product being used.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default

No default

Valid values

- CICS[®]
- MQ
- CB
- SF
- TUXEDO
- TOPEND
- blank or some other value (for UNIX and Windows; no other possible values for Solaris or SINIX)
- If applications are run in a WebSphere[®] Enterprise Server Edition CICS environment, this parameter should be set to "CICS"
- If applications are run in a WebSphere Enterprise Server Edition Component Broker environment, this parameter should be set to "CB"
- If applications are run in an IBM MQSeries[®] environment, this parameter should be set to "MQ"
- If applications are run in a BEA Tuxedo environment, this parameter should be set to "TUXEDO"
- If applications are run in an IBM San Francisco environment, this parameter should be set to "SF".

IBM WebSphere EJB and Microsoft Transaction Server users do not need to configure any value for this parameter.

If none of the products mentioned previously are being used, this parameter should not be configured but left blank.

In previous versions of IBM Db2 on Windows, this parameter contained the path and name of the DLL which contained the XA Transaction Manager's functions *ax_reg* and *ax_unreg*. This format is still supported. If the value of this parameter does not match any of the previously mentioned TP Monitor names, it will be assumed that the value is a library name which contains the *ax_reg* and *ax_unreg* functions. This is true for UNIX and Windows environments.

TXSeries® CICS Users: In previous versions of this product on Windows it was required to configure this parameter as "libEncServer:C" or "libEncServer:E". While this is still supported, it is no longer required. Configuring the parameter as "CICS" is sufficient.

MQSeries Users: In previous versions of this product on Windows it was required to configure this parameter as "mqmax". While this is still supported, it is no longer required. Configuring the parameter as "MQ" is sufficient.

Component Broker Users: In previous versions of this product on Windows it was required to configure this parameter as "somtrx1i". While this is still supported, it is no longer required. Configuring the parameter as "CB" is sufficient.

San Francisco Users: In previous versions of this product on Windows it was required to configure this parameter as "ibmsfDB2". While this is still supported, it is no longer required. Configuring the parameter as "SF" is sufficient.

The maximum length of the string that can be specified for this parameter is 19 characters.

It is also possible to configure this information in IBM Db2 Version 9.1's XA OPEN string. If multiple Transaction Processing Monitors are using a single Db2 instance, then it will be required to use this capability.

trust_allclnts - Trust all clients

This parameter and **trust_clntauth** are used to determine where users are validated to the database environment.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

YES [NO, YES, DRDAONLY]

This parameter is only active when the **authentication** parameter is set to CLIENT.

By accepting the default of YES for this parameter, all clients are treated as trusted clients. This means that the server assumes that a level of security is available at the client and the possibility that users can be validated at the client.

This parameter can only be changed to NO if the **authentication** parameter is set to CLIENT. If this parameter is set to NO, the untrusted clients must provide a userid

and password combination when they connect to the server. Untrusted clients are operating system platforms that do not have a security subsystem for authenticating users.

Setting this parameter to DRDAONLY protects against all clients except clients from Db2 for z/OS, Db2 for VM and VSE, and Db2 for OS/400®. Only these clients can be trusted to perform client-side authentication. All other clients must provide a user ID and password to be authenticated by the server.

When **trust_allclnts** is set to DRDAONLY, the **trust_clntauth** parameter is used to determine where the clients are authenticated. If **trust_clntauth** is set to CLIENT, authentication occurs at the client. If **trust_clntauth** is set to SERVER, authentication occurs at the client if no password is provided, and at the server if a password is provided.

trust_clntauth - Trusted clients authentication

This parameter specifies whether a trusted client is authenticated at the server or the client when the client provides a userid and password combination for a connection.

This parameter (and **trust_allclnts**) is only active if the **authentication** parameter is set to CLIENT. If a user ID and password are not provided, the client is assumed to have validated the user, and no further validation is performed at the server.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

CLIENT [CLIENT, SERVER]

If this parameter is set to CLIENT (the default), the trusted client can connect without providing a user ID and password combination, and the assumption is that the operating system has already authenticated the user. If it is set to SERVER, the user ID and password will be validated at the server.

When using the db2CfgSet API to set the database manager configuration parameter, the numeric value for CLIENT is 0. The numeric value for SERVER is 1.

util_impact_lim - Instance impact policy

This parameter allows the database administrator (DBA) to limit the performance degradation of a throttled utility on the workload.

Configuration type

Database manager

Applies to

- Database server with local clients
- Database server with local and remote clients

- Partitioned database server with local and remote clients

Parameter type
Configurable Online

Propagation class
Immediate

Default [range]
10 [1 - 100]

Unit of measure
Percentage of allowable impact on workload

If the performance degradation is limited, the DBA can then run online utilities during critical production periods, and be guaranteed that the performance impact on production work will be within acceptable limits.

For example, a DBA specifying a **util_impact_lim** (impact policy) value of 10 can expect that a throttled backup invocation will not impact the workload by more than 10 percent.

If **util_impact_lim** is 100, no utility invocations will be throttled. In this case, the utilities can have an arbitrary (and undesirable) impact on the workload. If **util_impact_lim** is set to a value that is less than 100, it is possible to invoke utilities in throttled mode. To run in throttled mode, a utility must also be invoked with a non-zero priority.

Recommendation: Most users will benefit from setting **util_impact_lim** to a low value (for example, between 1 and 10).

A throttled utility will usually take longer to complete than an unthrottled utility. If you find that a utility is running for an excessively long time, increase the value of **util_impact_lim**, or disable throttling altogether by setting **util_impact_lim** to 100.

wlm_dispatcher - Workload management dispatcher

This parameter enables (YES) or disables (NO) the Db2 workload management dispatcher. By default, an enabled dispatcher allows the setting of the CPU limits.

Configuration type
Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable Online

Propagation class
Immediate

Default [range]
NO [NO; YES]

When upgrading the Db2 database manager, the value of the **wlm_dispatcher** database manager configuration parameter is set to NO.

The workload management dispatcher provides CPU scheduling capabilities at the service class level in the Db2 database manager using shares-based allocation of CPU resources, or CPU limits, or both.

With the workload management dispatcher enabled, all work running in user and maintenance service classes is placed under the control of the dispatcher. When enabled, CPU limit settings are enforced by the dispatcher as the default case. In order to use shares-based allocation of CPU resources, the `wlm_disp_cpu_shares` database manager configuration parameter must be enabled.

wlm_disp_concur - Workload manager dispatcher thread concurrency

This parameter specifies how the Db2 workload manager (WLM) dispatcher sets the thread concurrency level. You can also manually set the thread concurrency level to a fixed value.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

COMPUTED [COMPUTED; *manually_set_value*]

When upgrading Db2 database manager, the value of the `wlm_disp_concur` database manager configuration parameter is COMPUTED.

COMPUTED

Db2 database manager computes a fixed thread concurrency level based upon the value of 4 times the number of logical CPUs available to the Db2 database manager.

manually_set_value

You can manually set the thread concurrency level to a fixed value (1 - 32767). The optimal value depends on the specific hardware used and the operating system level; generally, in the range of 2 to 4 times the number of logical CPUs on the host or LPAR.

Unit of measure

Number of concurrent threads

The setting of this database manager configuration parameter controls the number of threads that the WLM dispatcher allows to be dispatched to the operating system run queues in parallel. The value is set as a low multiple of the number of logical CPUs available to the Db2 database manager. In general, you can set the value to 4 times the number of available logical CPUs to take into account possible scheduling latencies that result when threads switch in and out of the active state. An optimal value is just large enough to ensure that there are an adequate numbers of threads for the Db2 database manager to fully use the CPUs on the

host or LPAR and no larger. This optimal value ensures maximum efficiency and gives the Db2 WLM dispatcher maximum control over CPU allocation.

wlm_disp_cpu_shares - Workload manager dispatcher CPU shares

This parameter enables (YES) or disables (NO) the control of CPU shares by the Db2 workload manager (WLM) dispatcher. By default, an enabled WLM dispatcher controls only CPU limits.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

NO [NO; YES]

When upgrading Db2 database manager, the value of the **wlm_disp_cpu_shares** database manager configuration parameter is NO.

If the value of the **wlm_dispatcher** database manager configuration parameter is set to YES and the value of the **wlm_disp_cpu_shares** database manager configuration parameter is set to NO, the WLM dispatcher can apply only CPU limits to the management of service classes.

If the value of the **wlm_dispatcher** database manager configuration parameter is set to YES and the value of the **wlm_disp_cpu_shares** database manager configuration parameter is set to YES, the WLM dispatcher can apply both CPU limits and CPU shares to the management of service classes. By default, all service classes are assigned 1000 hard CPU shares to ensure an equal division of CPU resources.

Table 8. Summary of required database manager configuration parameter settings for service class management by the Db2 WLM dispatcher

Service class management	Setting of wlm_dispatcher	Setting of wlm_disp_cpu_shares
None	NO	NO
CPU limits	YES	NO
CPU limits + CPU shares	YES	YES

wlm_disp_min_util - Workload manager dispatcher minimum CPU utilization

This parameter specifies the minimum amount of CPU utilization that is necessary for a service class to be included in the Db2 WLM-managed sharing of CPU resources.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Multimember database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

5 [0 to 100]

When upgrading Db2 database manager, the value of the **wlm_disp_min_util** database manager configuration parameter is 5.

Unit of measure

Percent

To illustrate the usage of this database manager configuration parameter with an example, suppose there are three service classes, A, B, and C, and each has 1000 shares of the CPU resources. In this example, the same result is obtained whether the service class shares are hard or soft CPU shares. Service classes A and B each have a CPU utilization that is greater than or equal to the 8% value set for the **wlm_disp_min_util** configuration parameter. Service class C has a 3% CPU utilization that is less than the 8% value set for the **wlm_disp_min_util** configuration parameter. In CPU share calculations, service class C is considered to not have any executing work. Therefore, only service classes A and B equally share the CPU resources and each receives a 50% share. When service class C begins to execute work to an extent that the CPU utilization is greater than or equal to the 8% value set for the **wlm_disp_min_util** configuration parameter, at this point service classes A, B, and C are now considered to equally share the CPU resources and each receives a 33.3% share.

In multimember database environments, it is the aggregate of the CPU utilizations of all the members on a host or LPAR that is compared to the **wlm_disp_min_util** configuration parameter to determine if the host or LPAR is included in the WLM-managed sharing of CPU resources.

Db2 database configuration parameters

Database configuration parameters specify, among other things, the amount of resources to allocate to a database. Values for many of the parameters can be changed to improve performance or increase capacity. Some parameters are for informational purposes only. This type of parameter affects Db2 at the database level.

alt_collate - Alternate collating sequence

This parameter specifies the collating sequence that is to be used for Unicode tables in a non-Unicode database.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type
Configurable

Default [range]
Null [IDENTITY_16BIT]

Until this parameter is set, Unicode tables and routines cannot be created in a non-Unicode database. Once set, this parameter cannot be changed or reset.

This parameter cannot be set for Unicode databases.

app_ctl_heap_sz - Application control heap size

This parameter has been deprecated since Version 9.5, but is still being used by pre-Version 9.5 data servers and clients. Any value specified for this configuration parameter is ignored by the database manager in Db2 Version 9.5 or later releases.

In Version 9.5, *app_ctl_heap_sz* has been replaced by the *appl_memory* configuration parameter.

Note: The following information applies only to pre-Version 9.5 data servers and clients.

Configuration type
Database

Parameter type
Configurable

Default [range]

Database server with local and remote clients

- 128 [1 - 64 000] when **intra_parallel** is not enabled
- 512 [1 - 64 000] when **intra_parallel** is enabled

Database server with local clients

On Linux and UNIX platforms:

- 128 [1 - 64 000] when **intra_parallel** is not enabled
- 512 [1 - 64 000] when **intra_parallel** is enabled

On Windows platforms:

- 64 [1 - 64 000] when **intra_parallel** is not enabled
- 512 [1 - 64 000] when **intra_parallel** is enabled

Partitioned database server with local and remote clients

512 [1 - 64 000]

Unit of measure
Pages (4 KB)

When allocated
When an application starts

When freed
When an application completes

For partitioned databases and for non-partitioned databases with intra-parallelism enabled, this parameter specifies the average size of the shared memory area allocated for an application. For non-partitioned databases where intra-parallelism is disabled, this is the maximum private memory that is allocated for the application control heap. There is one application control heap per connection per database partition.

The application control heap is required primarily for sharing information between agents working on behalf of the same request. Usage of this heap is minimal for non-partitioned databases when running queries with a degree of parallelism equal to 1.

This heap is also used to store descriptor information for declared temporary tables. The descriptor information for all declared temporary tables that have not been explicitly dropped is kept in the memory of the application control heap and cannot be dropped until the declared temporary table is dropped.

Recommendation: Initially, start with the default value. You might have to set the value higher if you are running complex applications, if you have a system that contains a large number of database partitions, or if you use declared temporary tables. The amount of memory needed increases with the number of concurrently active declared temporary tables. A declared temporary table with many columns has a larger table descriptor size than a table with few columns, so having a large number of columns in an application's declared temporary tables also increases the demand on the application control heap.

appgroup_mem_sz - Maximum size of application group memory set

This parameter has been deprecated since Version 9.5, but is still being used by pre-Version 9.5 data servers and clients. Any value specified for this configuration parameter will be ignored by the database manager in Db2 Version 9.5 or later releases.

In Version 9.5, it has been replaced by the **app1_memory** configuration parameter.

Note: The following information applies only to pre-Version 9.5 data servers and clients.

This parameter determines the size of the application group shared memory segment.

Configuration type
Database

Parameter type
Configurable

Default [range]

UNIX Database server with local clients (other than 32-bit HP-UX)
20 000 [1 - 1 000 000]

32-bit HP-UX

- Database server with local clients
- Database server with local and remote clients
- Partitioned database server with local and remote clients

10 000 [1 - 1 000 000]

Windows Database server with local clients

10 000 [1 - 1 000 000]

Database server with local and remote clients (other than 32-bit HP-UX)

30 000 [1 - 1 000 000]

Partitioned database server with local and remote clients (other than 32-bit HP-UX)

40 000 [1 - 1 000 000]

Unit of measure

Pages (4 KB)

Information that needs to be shared between agents working on the same application is stored in the application group shared memory segment.

In a partitioned database, or in a non-partitioned database with intrapartition parallelism enabled or concentrator enabled, multiple applications share one application group. One application group shared memory segment is allocated for the application group. Within the application group shared memory segment, each application will have its own application control heap, and all applications will share one application group shared heap.

The number of applications in one application group is calculated by:

$\text{appgroup_mem_sz} / \text{app_ctl_heap_sz}$

The application group shared heap size is calculated by:

$\text{appgroup_mem_sz} * \text{groupheap_ratio} / 100$

The size of each application control heap is calculated by:

$\text{app_ctl_heap_sz} * (100 - \text{groupheap_ratio}) / 100$

Recommendation: Retain the default value of this parameter unless you are experiencing performance problems.

appl_memory - Application Memory configuration parameter

The **appl_memory** configuration parameter specifies the size of the application memory set. The application memory size counts towards any **instance_memory** limit in effect.

The **appl_memory** setting must be large enough to accommodate the concurrent requirements of the following memory pools:

- The application group shared heap memory pool. This memory pool is a global work area for all applications and is not configurable.
- The application heap memory pool. This memory pool is configured per application and can be configured by using the **applheapsz** configuration parameter.
- The statement heap memory pool. This memory pool is configured per statement compilation and can be configured by using the **stmtheap** configuration parameter.
- The statistics heap memory pool. This memory pool is configured per RUNSTATS operation and can be configured by using the **stat_heap_sz** configuration parameter.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

- Configurable online (requires a database connection)
- Configurable by member in a Db2 pureScale environment and in partitioned database environments

Default [range]

Automatic [128 - 4 294 967 295]

- On 32-bit architectures, the default value is AUTOMATIC with an underlying value of 10000
- On 64-bit architectures, the default value is AUTOMATIC with an underlying value of 40000

Unit of measure

Pages (4 KB)

When allocated or committed**On Linux and UNIX operating systems**

The initial size is allocated on database activation. More memory is allocated as required.

On Windows operating systems

Memory is allocated as required. A minimal amount of application memory is committed on database activation. More memory is committed as required.

When freed

All application memory is freed when a database is deactivated. However, portions of allocated or committed memory are regularly released back to the operating system when these portions of memory are no longer in use.

On UNIX operating systems, after the initial application memory size is allocated during database activation, Db2 allocates more memory as needed to support dynamic memory requirements. Extra memory allocation is subject to any specified fixed size limit. All application memory is allocated as shared memory and is retained until the database deactivates. The total allocated shared memory counts towards only virtual memory usage. While this virtual memory does not require backing by real memory, virtual memory does require backing by swap or paging space on some operating systems. For details about operating system support, see the Operating System Support section.

On Windows operating systems, application memory is allocated as private memory. Application memory allocation is subject to any specified fixed size limit. Memory allocations no longer in use might be freed dynamically or retained for reuse. All outstanding memory allocations are freed when the database deactivates.

Committed memory is memory that is backed by the operating system. Allocated memory is committed as required by memory pools. Committed memory no longer required by memory pools is either cached to improve performance or released to the operating system. Memory is also released or decommitted as

necessary if the application memory size is decreased dynamically. All committed memory is released when the database deactivates.

It is recommended to leave **appl_memory** set to the default of AUTOMATIC. An insufficient fixed application memory setting results in various out of memory failures that are returned to applications. Setting a fixed memory value must be done only after thorough testing to determine peak requirements. Application memory is not tuned by the Self Tuning Memory Manager (STMM), but STMM tunes **database_memory**, if **database_memory** is enabled for self-tuning, to compensate for fluctuating application memory requirements.

Operating system support

Table 9. Operating system support

Operating System	Available support
AIX	Uses medium (64K) pages by default, which can benefit performance. Large (16MB) pages are also allowed on AIX only. ¹
HP-UX	Allocated shared memory requires backing by virtual swap.
Linux	Allocated shared memory counts towards the virtual shared memory limit (shmall).
Solaris	Allocated shared memory requires backing by virtual swap and counts towards any virtual memory limits.
Windows	No additional considerations for the Windows platform.
Note:	
1. The use of large pages (DB2_LARGE_PAGE_MEM) limits the ability of Db2 to release memory dynamically. It is recommended to configure a fixed appl_memory value when using large page memory.	

Monitoring

The application memory set can be monitored through the **MON_GET_MEMORY_SET** and **MON_GET_MEMORY_POOL** routines. For example, the following command:

```
db2 "select member, substr(db_name,1,10)as db_name, substr(memory_set_type,1,10) as set_type,
memory_set_size, memory_set_committed, memory_set_used, memory_set_used_hwm
from table(mon_get_memory_set('APPLICATION'))"
```

Returns the following information:

MEMBER	DB_NAME	SET_TYPE	MEMORY_SET_SIZE	MEMORY_SET_COMMITTED	MEMORY_SET_USED	MEMORY_SET_USED_HWM
0	SAMPLE	APPLICATION	154927	68616	67829	68616
0	TEST	APPLICATION	238092	123404	123404	123404

2 record(s) selected.

In this case, the application memory set is using 154927KB of **instance memory** (MEMORY_SET_SIZE) and 68616KB of **system memory** (MEMORY_SET_COMMITTED), of which 67829KB (MEMORY_SET_USED) is assigned to memory pools.

You can also monitor database memory using the **db2pd** utility:

```
db2pd -db <database_name> -memsets -mempools, db2pd -dbptnmem
```

applheapsz - Application heap size

The **applheapsz** configuration parameter refers to the total amount of application memory that can be consumed by the entire application.

In versions of Db2 prior to Version 9.5, the **applheapsz** database configuration parameter referred to the amount of application memory each individual database agent working for that application could consume.

With Version 9.5, this database configuration parameter has a default value of **AUTOMATIC**, meaning that it increases as needed until either the **appl_memory** limit is reached, or the **instance_memory** limit is reached. For partitioned database environments, Concentrator, or SMP configurations, this means that the **applheapsz** value used in previous releases might need to be increased under similar workloads, unless the **AUTOMATIC** setting is used.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]

32-bit platforms

Automatic [16 - 60 000]

64-bit platforms

Automatic [16 - 2 147 483 647]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

When an application associates with, or connects to, a database.

When freed

When the application disassociates or disconnects from the database.

Note: This parameter defines the maximum size of the application heap. One application heap is allocated per database application when the application first connects with the database. The heap is shared by all database agents working for that application. (In previous releases, each database agent allocated its own application heap.) Memory is allocated from the application heap as needed to process the application, up to the limit specified by this parameter. When set to **AUTOMATIC**, the application heap is allowed to grow as needed up to either the **appl_memory** limit for the database, or the **instance_memory** limit for the database partition. The entire application heap is freed when the application disconnects with the database.

The online changed value takes effect at an application connection boundary, that is, after it is changed dynamically, currently connected applications still use the old value, but all newly connected applications will use the new value.

archretrydelay - Archive retry delay on error

This parameter specifies the number of seconds to wait after a failed archive attempt before trying to archive the log file again.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Default [range]

20 [0 - 65 535]

Subsequent retries will only take affect if the value of the **numarchretry** database configuration parameter is at least 1.

auto_del_rec_obj - Automated deletion of recovery objects configuration parameter

This parameter specifies whether database log files, backup images, and load copy images should be deleted when their associated recovery history file entry is pruned.

Configuration type

Database

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

OFF [ON; OFF]

You can prune the entries in the recovery history file using the **PRUNE HISTORY** command or the db2Prune API. You can also configure the IBM Data Server database manager to automatically prune the recovery history file after each full database backup. If you set the **auto_del_rec_obj** database configuration parameter to ON, then the database manager will also delete the corresponding physical log files, backup images, and load copy images when it prunes the history file. The database manager can only delete recovery objects such as database logs, backup images, and load copy images when your storage media is disk, or if you are using a storage manager, such as the Tivoli Storage Manager. If the **logarchmeth1** parameter is set to LOGRETAIN and the **ARCHIVE LOG** command is issued, the log files will not be deleted by the prune even if entries appear in the history file and **auto_del_rec_obj** is set to ON.

auto_maint - Automatic maintenance

This parameter is the parent of all the other automatic maintenance database configuration parameters (**auto_db_backup**, **auto_tbl_maint**, **auto_runstats**, **auto_stmt_stats**, **auto_stats_views**, **auto_reorg**, and **auto_sampling**).

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

ON [ON; OFF]

When this parameter is disabled, all of its child parameters are also disabled, but their settings, as recorded in the database configuration file, do not change. When this parent parameter is enabled, recorded values for its child parameters take effect. In this way, automatic maintenance can be enabled or disabled globally.

By default, this parameter is set to ON.

You can enable or disable individual automatic maintenance features independently by setting the following parameters:

auto_db_backup

This automated maintenance parameter enables or disables automatic backup operations for a database. A backup policy (a defined set of rules or guidelines) can be used to specify the automated behavior. The objective of the backup policy is to ensure that the database is being backed up regularly. The backup policy for a database is created automatically when the Db2 Health Monitor first runs. By default, this parameter is set to OFF. To be enabled, this parameter must be set to ON, and its parent parameter must also be enabled.

auto_tbl_maint

This parameter is the parent of table maintenance parameters (**auto_runstats** and **auto_reorg**). When this parameter is disabled, all of its child parameters are also disabled, but their settings, as recorded in the database configuration file, do not change. When this parent parameter is enabled, recorded values for its child parameters take effect. In this way, table maintenance can be enabled or disabled globally.

By default, this parameter is set to ON.

auto_runstats

This automated table maintenance parameter enables or disables automatic table **RUNSTATS** operations for a database. A **RUNSTATS** policy (a defined set of rules or guidelines) can be used to specify the automated behavior. Statistics collected by the **RUNSTATS** utility are used by the optimizer to determine the most efficient plan for accessing the physical data. To be enabled, this parameter must be set to ON, and its parent parameters must also be enabled.

By default, this parameter is set to ON.

auto_stmt_stats

This parameter enables and disables the collection of real-time statistics. It is a child of the **auto_runstats** configuration parameter. This feature is enabled only if the parent, **auto_runstats** configuration parameter, is also enabled. For example, to enable **auto_stmt_stats**, set **auto_maint**, **auto_tbl_maint**, and **auto_runstats** to ON. To preserve the child value, the **auto_runstats** configuration parameter can be ON while the **auto_maint** configuration parameter is OFF. The corresponding Auto Runstats feature will still be OFF.

Assuming that both Auto Runstats and Auto Reorg are enabled, the settings are as follows:

Automatic maintenance	(AUTO_MAINT) = ON
Automatic database backup	(AUTO_DB_BACKUP) = OFF
Automatic table maintenance	(AUTO_TBL_MAINT) = ON
Automatic runstats	(AUTO_RUNSTATS) = ON
Real-time statistics	(AUTO_STMT_STATS) = ON
Statistical views	(AUTO_STATS_VIEWS) = OFF
Automatic sampling	(AUTO_SAMPLING) = OFF
Automatic reorganization	(AUTO_REORG) = ON

You can disable both Auto Runstats and Auto Reorg features temporarily by setting **auto_tbl_maint** to OFF. Both features can be enabled later by setting **auto_tbl_maint** back to ON. You do not need to issue **db2stop** or **db2start** commands to have the changes take effect.

By default, this parameter is set to ON.

auto_stats_views

This parameter enables and disables automatic statistic collection on statistical views. The statistics on statistical views are automatically maintained when this parameter is set to ON.

By default, this parameter is set to OFF.

auto_reorg

This automated table maintenance parameter enables or disables automatic table and index reorganization for a database. A reorganization policy (a defined set of rules or guidelines) can be used to specify the automated behavior. To be enabled, this parameter must be set to ON, and its parent parameters must also be enabled.

By default, this parameter is set to OFF.

auto_sampling

This parameter controls whether automatic statistics collection uses sampling when collecting statistics for a large table. To enable the automatic sampling, set **auto_maint**, **auto_tbl_maint**, **auto_runstats** and **auto_sampling** to ON. If the automatic statistics collection is enabled, page-level sampling is used and the sampling rate is determined automatically. Both data and index pages are sampled for base tables as opposed to statistical views, which do not have indexes.

By default, this parameter is set to ON.

auto_reval - Automatic revalidation and invalidation configuration parameter

This configuration parameter controls the revalidation and invalidation semantics.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

DEFERRED [IMMEDIATE, DISABLED, DEFERRED, DEFERRED_FORCE]

This configuration parameter is dynamic, meaning that a change in its value takes effect immediately. You do not have to reconnect to the database for the change to take effect.

If you create a new database, by default this configuration parameter is set to DEFERRED.

If you upgrade a database from Version 9.5, or earlier, **auto_reval** is set to DISABLED. The revalidation behavior is the same as in the previous releases.

If you set this parameter to IMMEDIATE it means that all dependent objects will be revalidated immediately after objects are invalidated. This applies to some DDL statements, such as ALTER TABLE, ALTER COLUMN, or CREATE OR REPLACE. The successful revalidation of the dependent objects does not rely on any other DDL changes; therefore, revalidation can be completed immediately.

If you set this parameter to DEFERRED, it means that all dependent objects are revalidated the next time that they are accessed. To reduce the impact of object revalidation on future access, once you have finished with the changes that cause object invalidation, it is strongly recommended that you run the SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS procedure to revalidate invalid objects affected and the **db2rbind** command to revalidate invalid packages.

Note that if you set this parameter either to IMMEDIATE or DEFERRED, and if any revalidation operation fails, the invalid dependent objects will remain invalid until the next time that they are accessed.

If you set this parameter to DEFERRED_FORCE it behaves the same way as when it is set to DEFERRED and an additional CREATE with error feature is enabled.

In some cases, the syntax that you explicitly specify might override the setting of **auto_reval**. For example, if you use the DROP COLUMN clause of the ALTER TABLE statement without specifying CASCADE or RESTRICT, the semantics are controlled by **auto_reval**. However, if you specify CASCADE or RESTRICT, the previous cascade or restrict semantics are used, overriding the new semantics specified by **auto_reval**.

autorestart - Auto restart enable

The **autorestart** configuration parameter determines if the database manager automatically initiates crash recovery when a user connects to a database that had previously terminated abnormally. If the **autorestart** configuration parameter is not set, the user must issue an explicit restart database command before they can connect to the database.

Configuration type

Database

Parameter type
Configurable Online

Propagation class
Immediate

Default [range]
On [On; Off]

When the **autorestart** parameter is set to ON, the next database connection attempt performs crash recovery automatically if necessary.

In a Db2 pureScale environment, the automatic restart behavior changes. If a database member fails, member crash recovery is automatically initiated for that member. In addition, if both cluster caching facilities fail at the same time, group crash recovery is automatically initiated. During member crash recovery, no connections are allowed against the database through that member. An SQL1015N error is returned if you try to connect to the database through a member that requires crash recovery. If crash recovery fails, another attempt at crash recovery is made. If the second attempt at crash recovery fails, an ALERT is set for the member on the host where recovery fails and restart light for that member is performed on a different host.

When the **autorestart** parameter is set to OFF, the automated member crash recovery and group crash recovery processes are inactive and must be performed manually, if required. The crash recovery processes can be initiated with the **RESTART DATABASE** command.

In a Db2 pureScale environment, if a member fails and cannot be restarted on its original host (also known as home host), Db2 cluster services restarts that member on one of the other available member hosts in the Db2 pureScale cluster. This is known as restart light processing. When a member is in restart light mode, you cannot directly connect to it, therefore no manual restart is possible. You must first set the **autorestart** parameter back to ON. The restart light member then automatically detects the change in the database configuration parameter and initiates crash recovery if needed.

avg_appls - Average number of active applications

This parameter is used by the query optimizer to help estimate how much buffer pool space will be available at run time for the access plan chosen.

Configuration type
Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class
Statement boundary

Default [range]
Automatic [1 - maxappls]

Unit of measure
Counter

Recommendation: When running the Db2 database product in a multi-user environment, particularly with complex queries and a large buffer pool, you might want the query optimizer to know that multiple query users are using your system so that the optimizer should be more conservative in assumptions of buffer pool availability.

When this parameter is set to `AUTOMATIC`, the parameter is updated to an appropriate value immediately, when the database configuration file is created, or when the database configuration file is reset.

Setting this parameter might help the optimizer model buffer pool usage more accurately. If you set this parameter manually, begin with a value of 2, regardless of the average number of applications that you run. After you assess the behavior of the optimizer and test the performance of your application at this setting, you can increase the value of the parameter in small increments. Continue to assess the behavior of the optimizer and test the performance of your application each time you increase the value of the parameter. Setting this parameter to a value that is too high might cause the optimizer to underestimate the amount of buffer pool space that is available to the query.

After you change the value of this parameter, consider rebinding applications by using the **REBIND PACKAGE** command.

backup_pending - Backup pending indicator

The **backup_pending** parameter indicates whether you need to do a full backup of the database before accessing it.

Configuration type

Database

Parameter type

Informational

This parameter is set to `ON` only if the database configuration is changed so that the database moves from being unrecoverable to recoverable. That is, initially both the **logarchmeth1** and **logarchmeth2** parameters were set to `OFF`, then either one or both of these parameters is set, and the update to the database configuration is accepted.

blk_log_dsk_ful - Block on log disk full

This parameter can be set to prevent disk full errors from being generated when the Db2 database system cannot create a new log file in the active log path.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

No [Yes; No]

Instead of generating a disk full error, the Db2 database system will attempt to create the log file every five minutes until it succeeds. After each attempt, the Db2 database system writes a message to the Administration Notification log. The only way that you can confirm that your application is hanging because of a log disk full condition is to monitor the Administration Notification log. Until the log file is successfully created, any user application that attempts to update table data will not be able to commit transactions. Read-only queries might not be directly affected; however, if a query needs to access data that is locked by an update request, or a data page that is fixed in the buffer pool by the updating application, read-only queries will also appear to hang.

Setting **blk_log_dsk_ful** to yes causes applications to hang when the Db2 database system encounters a log disk full error, thus allowing you to resolve the error and allowing the transaction to complete. You can resolve a full log disk situation by either increasing the file system capacity or by removing extraneous files on the same file system.

If **blk_log_dsk_ful** is set to no, then a transaction that receives a log disk full error will fail and be rolled back.

Set **blk_log_dsk_ful** to yes when using infinite logging; that is, when the database configuration parameter **logsecond** is set to -1. With infinite logging, the database might go offline in some situations if a transaction causes a log disk full error.

blocknonlogged - Block creation of tables that allow non-logged activity

This parameter specifies whether the database manager will allow tables to have the NOT LOGGED or NOT LOGGED INITIALLY attributes activated.

Configuration type

Database

Parameter type

Configurable Online

Configurable by member in a Db2 pureScale environment

Default [range]

No [Yes, No]

By default, **blocknonlogged** is set to NO: non-logged operations are permitted and you gain the performance benefits associated with reduced logging. There are some potential drawbacks associated with this configuration, however, particularly in high availability disaster recovery (HADR) database environments. Db2 HADR database environments use database logs to replicate data from the primary database to the standby database. Non-logged operations are allowed on the primary database, but are not replicated to the standby database. If you perform any non-logged operations on the primary database, the standby database must be reinitialized. For example, you can use online split mirrors or suspended I/O support to resynchronize the standby database after non-logged operations.

Usage notes

- If **blocknonlogged** is set to YES, then CREATE TABLE and ALTER TABLE statements will fail if one of the following conditions is true:
 - The NOT LOGGED INITIALLY parameter is specified.
 - The NOT LOGGED parameter is specified for a LOB column.

- A CLOB, DBCLOB, or BLOB column is defined as not logged.
- If **blocknonlogged** is set to YES, then the LOAD command fails if the following situations exist:
 - You specify the NONRECOVERABLE option.
 - You specify the COPY NO option.

CF self-tuning memory database configuration parameter

In a Db2 pureScale environment, this read-only database configuration parameter indicates whether CF self-tuning memory is enabled or not.

Configuration type

Database

Parameter type

Informational

Default [range]

ON [ON, OFF]

In the Db2 Cancun Release and later releases, this read-only parameter is set ON when registry variable **DB2_DATABASE_CF_MEMORY** is set to *AUTO* or unset, and, these database configuration parameters are set to *AUTOMATIC*: **cf_gbp_sz**, **cf_lock_sz**, and **cf_sca_sz**.

When set ON, CF self-tuning memory is enabled.

cf_catchup_trgt - Target for catch up time of secondary cluster caching facility configuration parameter

This configuration parameter determines the target time in minutes for completing the catch up to bring a newly added or newly restarted cluster caching facility into peer state with an existing primary cluster caching facility .

Configuration type

Database

Parameter type

Configurable online

Default [range]

15 [1 - 600]

Unit of measure

Minutes since the secondary cluster caching facility was started

This parameter sets a target time for members to bring a newly added secondary cluster caching facility into peer state with the primary cluster caching facility.

Once it is in peer state, the secondary cluster caching facility is able to take over the role of the primary cluster caching facility if the primary cluster caching facility fails or is shut down for maintenance.

A lower setting for **cf_catchup_trgt** causes members to aggressively perform catch up activity, thus minimizing the window during which a Db2 pureScale instance is running without a secondary cluster caching facility that is ready to take over the role of the primary. A lower setting has a larger impact on database transactions and the overall performance of the system since more system resources (for example, I/O bandwidth) are used for catch up activity.

Note: Although the default setting for **cf_catchup_trgt** balances system performance with high availability, the following rule of thumb can be used to tune this parameter:

- If availability is more important than performance, use a lower setting for **cf_catchup_trgt**.
- If performance is more important than availability, use a higher setting for **cf_catchup_trgt**.

cf_db_mem_sz - Database memory configuration parameter

This parameter controls the total memory limit for the cluster caching facility, also designated as the CF, for this database.

Configuration type

Database

Applies to

Applies to a Db2 pureScale instance only.

- Database server with local and remote clients

Parameter type

Configurable online

Default [range]

AUTOMATIC [32768 - 4 294 967 295]

Unit of measure

Pages (4 KB)

The sum of the cluster caching facility structure memory limits for the **cf_gbp_sz**, **cf_lock_sz**, and **cf_sca_sz** parameters must be less than the CF structure memory limit for the **cf_db_mem_sz** parameter. Automatic memory tuning between CF resources such as the group buffer pool (GBP), shared communication area (SCA), and lock structures from within the same database is bound by the **cf_db_mem_sz** parameter.

Usage notes

- Set the value of the **numdb** configuration parameter higher than or equal to the total number of databases on the instance. Additionally, set the value of **DB2_DATABASE_CF_MEMORY** to a value that allows for every database on this instance, whether normally inactive or active, to be active at the same time.
- When manually setting the value of **cf_db_mem_sz**, the value must be less than the total CF server memory. The total CF server memory is controlled by the **cf_mem_sz** database manager configuration parameter.
- If the previous value of the **cf_db_mem_sz** parameter was set to AUTOMATIC, then the current value can be determined by using the **SHOW DETAIL** option of the **GET DATABASE CONFIGURATION** command.
- Use of the **cf_db_mem_sz** configuration parameter must be coordinated with the **DB2_DATABASE_CF_MEMORY** registry variable and the **numdb** configuration parameter.

Restrictions

- There is an additional 3840 4k pages taken from the **cf_mem_sz** parameter to be used internally by the CF. Additionally, there is an additional 256 4k pages reserved from **cf_mem_sz** for every active database in the instance. These additional pages only needs to be taken into consideration when setting **cf_mem_sz** manually.

- If altering the fixed value for this parameter to a new value that is *lower* than the current value, the new value must be greater than the sum of the memory sizes for the **cf_gbp_sz**, **cf_lock_sz**, and **cf_sca_sz** structure parameters or the operation will fail. To avoid this problem, lower the individual structure memory sizes before attempting to lower this parameter.
- If altering the fixed value for this parameter to a new value that is *higher* than the current value, there is an upper limit imposed, in addition to the total CF server memory defined by the **cf_mem_sz** parameter. Typically, this upper limit is defined by a memory buffer of between 3600 and 5000 pages that is required between the **cf_db_mem_sz** and **cf_mem_sz** parameter values. The value of the **cf_db_mem_sz** parameter should not exceed the difference of **cf_mem_sz** and this memory buffer.

cf_gbp_sz - Group buffer pool configuration parameter

This parameter determines the memory size used by the cluster caching facility, also known as CF, for group buffer pool (GBP) usage for this database.

Configuration type

Database

Applies to

Applies to a Db2 pureScale instance only.

- Database server with local and remote clients
- Client
- Database server with local clients

Parameter type

Configurable online

Default [range]

AUTOMATIC [AUTOMATIC, 4096 - 4294967296]

Unit of measure

Pages (4 KB)

When allocated

When the database is first connected or activated on any Db2 member

When freed

When the database is deactivated on the last Db2 member

There are two main types of resources in the GBP, directory entries and data elements. A directory entry stores metadata information pertaining to a page. A data element stores the actual page data.

Db2 members continue to cache pages in their own local buffer pools. The GBP is used only by the local buffer managers to maintain intersystem buffer coherency and cannot be directly referenced by Db2 EDUs running on any Db2 member.

cf_lock_sz - CF Lock manager configuration parameter

This parameter determines the memory size used by the CF for locking usage for this database.

Configuration type

Database

Applies to

Applies to a Db2 pureScale instance only.

- Database server with local and remote clients

Parameter type

Configurable online

Default [range]

AUTOMATIC [AUTOMATIC, 4096 - 1073741824]

Unit of measure

Pages (4 KB)

When allocated

When the database is first activated on any member.

Note: Memory is allocated on the CF and not on any of the members. If there is more than one CF then each CF allocates the same amount of memory.

When freed

When the database is deactivated on all members.

This is a global configuration parameter, meaning that the same value is used for every member in the Db2 pureScale instance.

Note that the memory on the CF is only allocated once; when the database is first activated on any of the members. You can monitor the consumption of the CF lock memory using the **current_cf_lock_size** monitor element.

cf_sca_sz - Shared communication area configuration parameter

This Shared Communication Area (SCA) configuration parameter determines the memory size used by the SCA in the cluster caching facility (CF). The SCA is a per database entity and contains database wide control block information for tables, indexes, table spaces, and catalogs.

Configuration type

Database

Applies to

Db2 pureScale environment

Parameter type

Configurable online

Default [range]

AUTOMATIC [AUTOMATIC, 2048 - 1073741824]

Unit of measure

Pages (4 KB)

When allocated

When the database is first activated on any Db2 member

When freed

When the database is dropped or when the CF server is stopped

When you set the **cf_sca_sz** parameter to AUTOMATIC (default), the memory value is computed by the Db2 database manager during the first database activation on any member to a value that is sufficient for basic database operations. If you want to configure the exact memory size for the SCA, you can set the **cf_sca_sz** parameter to a fixed numeric value.

You can obtain the current value for the SCA size by running the **GET DB CFG SHOW DETAIL** command. If the **cf_sca_sz** parameter is set to **AUTOMATIC**, the **SHOW DETAIL** clause displays the computed value as well as the allocated value for the SCA.

If the SCA memory is fully consumed by database operations, an out of memory error message is returned. In this case, you can increase the size of the SCA by setting a higher value for the **cf_sca_sz** parameter.

See “Configuring cluster caching facility memory for a database” for more detail.

catalogcache_sz - Catalog cache size

This parameter specifies the maximum space in pages that the catalog cache can use from the database heap.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

32-bit platforms

-1 [**maxapps***5, 8 - 524 288]

64-bit platforms

-1 [**maxapps***5, 8 - 2 147 483 647]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

When the database is initialized

When freed

When the database is shut down

This parameter is allocated out of the database shared memory, and is used to cache system catalog information. In a partitioned database system, there is one catalog cache for each database partition.

Caching catalog information at individual database partitions allows the database manager to reduce its processing time by eliminating the need to access the system catalogs (or the catalog node in a partitioned database environment) to obtain information that has previously been retrieved. The use of the catalog cache can help improve the overall performance of:

- Binding packages and compiling SQL and XQuery statements
- Operations that involve checking database-level privileges, routine privileges, global variable privileges and role authorizations
- Applications that are connected to non-catalog nodes in a partitioned database environment

By taking the default (-1) in a server or partitioned database environment, the value used to calculate the page allocation is five times the value specified for the **maxappls** configuration parameter. The exception to this occurs if five times **maxappls** is less than 8. In this situation, the default value of -1 will set **catalogcache_sz** to 8.

Recommendation: Start with the default value and tune it by using the database system monitor. When tuning this parameter, you should consider whether the extra memory being reserved for the catalog cache might be more effective if it was allocated for another purpose, such as the buffer pool or package cache.

Tuning this parameter is particularly important if a workload involves many SQL or XQuery compilations for a brief period of time, with few or no compilations thereafter. If the cache is too large, memory might be wasted holding copies of information that will no longer be used.

In an partitioned database environment, consider if the **catalogcache_sz** at the catalog node needs to be set larger since catalog information that is required at non-catalog nodes will always first be cached at the catalog node.

The **cat_cache_lookups** (catalog cache lookups), **cat_cache_inserts** (catalog cache inserts), **cat_cache_overflows** (catalog cache overflows), and **cat_cache_size_top** (catalog cache high water mark) monitor elements can help you determine whether you should adjust this configuration parameter.

Note: The catalog cache exists on all nodes in a partitioned database environment. Since there is a local database configuration file for each node, each node's **catalogcache_sz** value defines the size of the local catalog cache. In order to provide efficient caching and avoid overflow scenarios, you need to explicitly set the **catalogcache_sz** value at each node and consider the feasibility of possibly setting the **catalogcache_sz** on non-catalog nodes to be smaller than that of the catalog node; keep in mind that information that is required to be cached at non-catalog nodes will be retrieved from the catalog node's cache. Hence, a catalog cache at a non-catalog node is like a subset of the information in the catalog cache at the catalog node.

In general, more cache space is required if a unit of work contains several dynamic SQL or XQuery statements or if you are binding packages that contain a large number of static SQL or XQuery statements.

chngpgs_thresh - Changed pages threshold

This parameter specifies the level (percentage) of changed pages at which the asynchronous page cleaners will be started, if they are not currently active.

Configuration type

Database

Parameter type

- Configurable
- Configurable by member in a Db2 pureScale environment

Default [range]

60 [5 - 99]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Percentage

Asynchronous page cleaners will write changed pages from the buffer pool (or the buffer pools) to disk before the space in the buffer pool is required by a database agent. As a result, database agents should not have to wait for changed pages to be written out so that they might use the space in the buffer pool. This improves overall performance of the database applications.

When the page cleaners are started, they will build a list of the pages to write to disk. Once they have completed writing those pages to disk, they will become inactive again and wait for the next trigger to start.

When the **DB2_USE_ALTERNATE_PAGE_CLEANSING** registry variable is set (that is, the alternate method of page cleaning is used), the **chnpggs_thresh** parameter has no effect, and the database manager automatically determines how many dirty pages to maintain in the buffer pool.

Recommendation: For databases with a heavy update transaction workload, you can generally ensure that there are enough clean pages in the buffer pool by setting the parameter value to be equal-to or less-than the default value. A percentage larger than the default can help performance if your database has a small number of very large tables.

codepage - Code page for the database

This parameter shows the code page that was used to create the database. The **codepage** parameter is derived based on the **codeset** parameter.

Configuration type

Database

Parameter type

Informational

codeset - Codeset for the database

This parameter shows the codeset that was used to create the database. Codeset is used by the database manager to determine **codepage** parameter values.

Configuration type

Database

Parameter type

Informational

collate_info - Collating information

This parameter determines the database's collating sequence. For a language-aware collation or locale-sensitive UCA collation, the first 256 bytes contain the string representation of the collation name (for example, "SYSTEM_819_US").

This parameter can only be displayed using the db2CfgGet API. It **cannot** be displayed through the command line processor.

Configuration type

Database

Parameter type

Informational

This parameter provides 260 bytes of database collating information. The first 256 bytes specify the database collating sequence, where byte “n” contains the sort weight of the code point whose underlying decimal representation is “n” in the code page of the database.

The last 4 bytes contain internal information about the type of the collating sequence. The last four bytes of the parameter is an integer. The integer is sensitive to the endian order of the platform. The possible values are:

- **0** - The sequence contains non-unique weights
- **1** - The sequence contains all unique weights
- **2** - The sequence is the identity sequence, for which strings are compared byte for byte.
- **3** - The sequence is NLSCHAR, used for sorting characters in a TIS620-1 (code page 874) Thai database.
- **4** - The sequence is IDENTITY_16BIT, which implements the “CESU-8 Compatibility Encoding Scheme for UTF-16: 8-bit” algorithm as specified in the Unicode Technical Report #26 available at the Unicode Technical Consortium website at <http://www.unicode.org>
- **X'8001'** - The sequence is UCA400_NO, which implements the Unicode Collation Algorithm (UCA) based on the Unicode Standard version 4.0.0, with normalization implicitly set to ON.
- **X'8002'** - The sequence is UCA400_LTH, which implements the Unicode Collation Algorithm (UCA) based on the Unicode Standard version 4.0.0, and sorts all Thai characters as per the Royal Thai Dictionary order.
- **X'8003'** - The sequence is UCA400_LSK, which implements the Unicode Collation Algorithm (UCA) based on the Unicode Standard version 4.0.0, and sorts all Slovakian characters properly.

Note:

- For a language-aware collation or locale-sensitive UCA collation, the first 256 bytes contain the string representation of the collation name.
- Collations based on the Unicode Collation Algorithm of the Unicode Standard version 4.0.0 have been deprecated in Version 10.1 and might be removed in a future release.

If you use this internal type information, you need to consider byte reversal when retrieving information for a database on a different platform.

You can specify the collating sequence at database creation time.

connect_proc - Connect procedure name database configuration parameter

This database configuration parameter allows you to input or update a two-part connect procedure name that will be executed every time an application connects to the database.

Configuration type

Database

Parameter type

Configurable Online

Configurable by member in a Db2 pureScale environment

Default

NULL

The following connect procedure conventions must be followed, otherwise an error is returned.

- The non-zero length string must specify a two-part procedure name (that is, [schema name].[procedure name])
- The connect procedure name (both schema and procedure name) can only contain the following characters:
 - A-Z
 - a-z
 - _ (underscore)
 - 0-9
- In addition, the schema and procedure name need to follow the rules of an ordinary identifier.

Once the **connect_proc** parameter is configured to a non-zero length value, the server will implicitly execute the procedure specified on every new connection.

Usage Notes

- A connection to the database is required when updating this parameter. However, unsetting the parameter does not require a connection if the database is deactivated.
- The **connect_proc** parameter can only be set using the **IMMEDIATE** option of the **UPDATE DATABASE CONFIGURATION** command. The **DEFERRED** option cannot be used when setting the **connect_proc** parameter.
- Only a procedure with exactly zero parameters can be used as a connect procedure. No other procedure sharing the same two-part name can exist in the database as long as the **connect_proc** parameter is set.
- The connect procedure must exist in the database before updating the **connect_proc** parameter. The **UPDATE DATABASE CONFIGURATION** command will fail with an error if the connect procedure with zero parameters does not exist in the database or if there is more than one procedure with the same name.
- Use the same connect procedure on all partitions in a data-partitioned environment.
- Before you perform a **RESTORE DATABASE** on an existing image of the database, you must reset the **connect_proc** parameter to NULL. If the **connect_proc** is not set to NULL, you might encounter ERROR SQL0440N when you attempt a connection or rollforward command. To avoid this error, you must update the **connect_proc** parameter to NULL by using the following command:
db2 update db cfg for <DATABASE> using connect_proc NULL

country/region - Database territory code

This parameter shows the *territory* code used to create the database.

Configuration type

Database

Parameter type

Informational

cur_commit - Currently committed configuration parameter

This parameter controls the behavior of cursor stability (CS) scans.

Configuration type

Database

Parameter type

- Configurable
- Configurable by member in a Db2 pureScale environment

Default [range]

ON [ON, AVAILABLE, DISABLED]

For new databases, the default is set to ON. When the default is set to ON your query will return the currently committed value of the data at the time when your query is submitted.

During database upgrade from V9.5 or earlier, the **cur_commit** configuration parameter is set to DISABLED to maintain the same behavior as in previous releases. If you want to use currently committed on cursor stability scans, you need to set the **cur_commit** configuration parameter to ON after the upgrade.

You can explicitly set the **cur_commit** configuration parameter to AVAILABLE. Once you set this parameter, you need to explicitly request for currently committed behavior to see the results that are currently committed.

Note: Three registry variables **DB2_EVALUNCOMMITTED**, **DB2_SKIPDELETED**, and **DB2_SKIPINSERTED** are affected by currently committed when cursor stability isolation level is used. These registry variables are ignored when **USE CURRENTLY COMMITTED** or **WAIT FOR OUTCOME** are specified explicitly on the **BIND** or at statement prepare time.

Note: Performance considerations might be applicable in a database where there are significant lock conflicts when using currently committed. The committed version of the row is retrieved from the log, and will perform better and avoid log disk activity when the log record is still in the log buffer. Therefore, to improve the performance of retrieving previously committed data, you might consider an increase to the value of the **logbufsz** parameter.

database_consistent - Database is consistent

This parameter indicates whether the database is in a consistent state.

Configuration type

Database

Parameter type

Informational

YES indicates that all transactions have been committed or rolled back so that the data is consistent. If the system “crashes” while the database is consistent, you do not need to take any special action to make the database usable.

NO indicates that a transaction is pending or some other task is pending on the database and the data is not consistent at this point. If the system “crashes” while the database is not consistent, you will need to restart the database using the **RESTART DATABASE** command to make the database usable.

database_level - Database release level

This parameter indicates the release level of the database manager which can use the database.

Configuration type

Database

Parameter type

Informational

In the case of an incomplete or failed database upgrade, this parameter will reflect the release level of the database before the upgrade and might differ from the **release** parameter (the release level of the database configuration file). Otherwise the value of **database_level** will be identical to value of the **release** parameter.

database_memory - Database shared memory size

The **database_memory** configuration parameter specifies the size of the database memory set. The database memory size counts towards any instance memory limit in effect.

The setting must be large enough to accommodate the following configurable memory pools: bufferpools, the database heap, the locklist, the utility heap, the package cache, the catalog cache, the shared sort heap, and an additional minimum overflow area of 5%.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

- Configurable Online (requires a database connection)

Note: Requires a database connection. Dynamic changes are limited for pinned memory and large page configurations.

- Configurable by member in a Db2 pureScale environment and in partitioned database environments

Default [range]

AUTOMATIC [AUTOMATIC, COMPUTED, 0 - 4 294 967 295]

Unit of measure

Pages (4 KB)

When allocated or committed

On Linux and UNIX operating systems

The initial size is allocated on database activation. Additional memory is allocated as required.

On Windows operating systems

Memory is allocated as required.

Memory for bufferpools, the locklist, and basic infrastructure is committed during database activation. Additional memory is committed as required.

When freed

All database memory is freed when a database is deactivated. The Self Tuning Memory Manager (STMM) releases memory back to the operating system as it decreases the database memory size. Additional releasing of memory is subject to the **db_mem_thresh** configuration parameter.

On UNIX operating systems, after allocating the initial database memory size on database activation, Db2 allocates additional memory as needed to support dynamic requirements. Additional memory allocation is subject to any fixed size limit. All database memory is allocated as shared memory and is retained until the database deactivates. The total allocated shared memory counts only towards virtual memory usage. While this does not require backing by real memory, virtual memory does require backing by swap or paging space on some operating systems. On Windows operating systems, database memory is allocated as private memory as required, subject to any fixed size limit. Allocations no longer in use might be freed dynamically or retained for reuse. All outstanding memory allocations are freed when the database deactivates. For details about operating system support, see the Operating System Support section.

Committed memory is memory that is backed by the operating system. With the exception of pinned memory, which includes large page configurations, allocated memory is committed as required by memory pools. Committed memory no longer required by memory pools is either cached to improve performance or released (decommitted) back to the operating system. The action taken is subject to the **db_mem_thresh** configuration parameter. Memory is also released or decommitted as necessary when the **database_memory** size is reduced, such as by STMM. All committed memory is released when the database deactivates.

The database memory size counts towards instance memory usage. The database memory overflow area is equivalent to cached instance memory for the database memory consumer. Database memory overflow can be dynamically reduced as necessary to accommodate the requirements of other memory areas used by the database instance. This is done automatically when an instance memory limit is in effect.

The configured sizes of the underlying memory pools are reserved from the database memory size. The remaining database memory is considered overflow memory. Memory pools are normally allowed to use any available overflow, also known as unreserved memory.

The Self Tuning Memory Manager (STMM) is a feature that you can use to automatically configure key memory areas. You can enable STMM to tune the overall database memory size as well as the following areas within database memory:

- Bufferpools
- The locklist
- The shared sort heap (If SHEAPTHRES is 0)
- The package cache

The size of the overflow area is dependent on the size of database memory. Large database memory sizes require less overflow and smaller sizes require more overflow, up to a maximum of 10%. The overflow amount is calculated during database activation and on an ongoing basis as part of STMM overflow tuning.

Table 10. Comparison of database memory size and overflow

Database memory size	Overflow target
64 GB or less	10%
64 - 96 GB	9%
96 - 156 GB	8%
156 - 266 GB	7%
266 - 493 GB	6%
493 GB or more	5%

The behaviour of values that you can assign to **database_memory** are as follows.

AUTOMATIC

When **database_memory** is set to AUTOMATIC, the initial database memory size is calculated based on the underlying configuration requirements. This includes:

- Bufferpools
- The database heap
- The locklist
- The utility heap
- The package cache
- The catalog cache
- The shared sort heap, if it is enabled
- The overflow area

The AUTOMATIC setting allows database memory to grow beyond its initial size if there are unforeseen requirements beyond what the overflow provides. Dynamic configuration changes made manually or by STMM to individual memory pools also adjust the database memory size by a corresponding amount.

If STMM is enabled (The value of **SELF_TUNING_MEM** is ON), STMM controls the overall database memory size. STMM takes into account the underlying configuration requirements, including overflow, and the performance benefits of acquiring additional available memory. Depending on the **instance_memory** setting, STMM tunes database memory to avoid shortages of system and instance memory. A percentage of memory, that you can control with the **DB2_MEM_TUNING_RANGE** variable, is left available to satisfy volatile requirements. When a database is activated, if there is insufficient system or instance memory to support the starting configuration, any memory areas tuned by STMM are reduced to accommodate existing memory constraints. This action is subject to enforced minimum sizes.

Fixed Value

You can assign a specific value to the **database_memory** configuration parameter. However, the specified value should be large enough to support the minimum configuration requirements. This includes:

- Bufferpools
- The database heap
- The locklist
- The utility heap
- The package cache

- The catalog cache
- The shared sort heap, if it is enabled
- A 5% minimum overflow area

If the assigned value is too small, the higher minimum size required to support the configuration is allocated. Allocated database memory cannot exceed the fixed setting or higher minimum size. However, the database memory setting can still be increased dynamically or changed to AUTOMATIC. Dynamic configuration increases to memory pools only succeed if sufficient overflow is available.

If STMM is enabled (The value of **SELF_TUNING_MEM** is ON), only the underlying memory pools and overflow are tuned. At the time of database activation, if a fixed setting exists that is too small to support the current configuration, the individual areas tuned by STMM are reduced to accommodate existing memory constraints. This action is subject to enforced minimum sizes. It is recommended to leave at least half of a fixed **database_memory** configuration available for STMM tuning. For example, if manually setting the size of the bufferpool areas, avoid having the bufferpools consume more than half of a fixed **database_memory** setting. Doing so might constrain STMM tuning capabilities, resulting in suboptimal performance and symptoms related to constrained memory resources.

COMPUTED

COMPUTED is a legacy setting. Its behaviour is similar to AUTOMATIC when STMM is not enabled. When STMM is enabled, the behaviour of COMPUTED is similar to that of a fixed value.

Operating system support

Table 11. Operating system support

Operating System	Available support
AIX	Uses medium (64K) pages by default, which can benefit performance. AIX supports pinned memory and large/huge (16MB/16GB) pages. ¹
HP-UX	Allocated shared memory requires backing by virtual swap. HP-UX supports pinned memory. ¹
Linux	Allocated shared memory counts towards the virtual shared memory limit (shmall). Linux supports pinned memory and large (2MB) pages. ¹
Solaris	Allocated shared memory requires backing by virtual swap and counts towards any virtual memory limits. Solaris supports pinned ISM memory and large pages. ²
Windows	Supports large (2MB) pages. ¹

Table 11. Operating system support (continued)

Operating System	Available support
<p>Note:</p> <ol style="list-style-type: none"> 1. The use of memory pinning (DB2_PINNED_BP) and large/huge pages (DB2_LARGE_PAGE_MEM) limits the ability to release memory dynamically. STMM performs limited tuning in this type of environment, and does not attempt to adjust the overall database memory size. It is recommended to configure a fixed database_memory value when using STMM under these configurations, which allows STMM to make optimal use of a consistent database memory sizing. 2. When database_memory is set to AUTOMATIC on the Solaris operating system, the database manager uses pageable memory for database shared memory. This allows database memory configuration adjustments to be fully dynamic, and STMM tunes the overall database memory size. On SPARC architectures, the database manager attempts to use 64 KB memory pages if available. Otherwise 8 KB memory pages are used. On x64 architectures, the database manager uses 4KB memory pages. When using the fixed or COMPUTED options for database memory, ISM (pinned) memory is allocated. In this case Solaris chooses the largest appropriate page size, which might improve performance. However, dynamic database memory configuration changes is limited, and STMM does not attempt to tune the overall database memory size. Changes between AUTOMATIC and the other database_memory options cannot be performed dynamically on Solaris operating systems. 	

Monitoring

The database memory set can be monitored through the **MON_GET_MEMORY_SET** and **MON_GET_MEMORY_POOL** routines. For example, the following command:

```
db2 "select member, substr(db_name,1,10)as db_name, substr(memory_set_type,1,10) as set_type,
memory_set_size, memory_set_committed, memory_set_used, memory_set_used_hwm
from table(mon_get_memory_set('DATABASE','',-1))"
```

Returns the following information:

MEMBER	DB_NAME	SET_TYPE	MEMORY_SET_SIZE	MEMORY_SET_COMMITTED	MEMORY_SET_USED	MEMORY_SET_USED_HWM
0	SAMPLE	DATABASE	154927	68616	67829	68616
0	TEST	DATABASE	238092	123404	123404	123404

2 record(s) selected.

In this case, the database memory set is using 154927KB of **instance_memory** (**MEMORY_SET_SIZE**) and 68616KB of system memory (**MEMORY_SET_COMMITTED**), of which 67829KB (**MEMORY_SET_USED**) is assigned to memory pools.

You can also monitor database memory using the **db2pd** utility:

```
db2pd -db <database_name> -memsets -mempools, db2pd -dbptnmem
```

dbheap - Database heap

You can use this parameter to limit the maximum amount of memory allocated for the database heap. Additional memory is automatically added for critical memory requirements.

By default, **dbheap** is set to **AUTOMATIC**, meaning that the size of the database heap can increase as needed. This increase is subject to **database_memory**, **instance_memory**, and system memory limits.

Configuration type

Database

Parameter type

Configurable online (requires a database connection)

Configurable by a member in a Db2 pureScale environment or partitioned database environment

Propagation class

Immediate

Default [range]**On Linux and UNIX operating systems**

AUTOMATIC [AUTOMATIC, 32 - 2 147 483 647]

- The default value is AUTOMATIC with an underlying value of 1200.

On Windows operating systems

AUTOMATIC [AUTOMATIC, 32 - 2 147 483 647]

- On 32-bit architectures, the default value for a standalone server is AUTOMATIC with an underlying value of 300.
- On 64-bit architectures, the default value is AUTOMATIC with an underlying value of 600.

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

The underlying or fixed value of **dbheap** and an internal memory allowance for critical requirements is reserved from database memory at database activation. The database heap is allocated as required.

When freed

Memory allocations are freed when they are no longer needed. All allocated memory in the database heap is freed when the database is deactivated.

There is one database heap per database, and it is used for a variety of purposes critical to the support of database-wide activities. Many known requirements, such as the log buffer, are evaluated at database activation time. These internal requirements, along with the underlying or fixed configured **dbheap** value, contribute towards the initial database memory sizing and constitute a reservation of database memory for the database heap. Additional requirements which are not part of the internal calculation include a cache for table metadata, which accumulates as tables are accessed.

The initial database heap reservation includes both the configured value and the internally calculated requirement. This initial reservation is an enforced hard limit if the database heap is set to a fixed value. When the **dbheap** parameter is left at the default AUTOMATIC setting, the database heap is allowed to grow beyond the initial reservation and use any remaining database memory overflow. In addition, if the **database_memory** parameter is set to AUTOMATIC, then the database heap can grow further by automatically increasing the **database_memory** size.

The database heap requirements are dependent on a variety of factors including database design, application activity, and configuration. It is recommended that you leave the setting at AUTOMATIC with a low underlying value, such as the default

value. The **dbheap** setting changes to **AUTOMATIC** when you migrate to Db2 Version 9.5 or higher from earlier releases, or when you apply the recommendations from the Db2 Configuration Advisor.

Some of the key memory requirements are:

- A log buffer of size **LOGBUFSZ**, which is allocated at database activation. This is part of the internal database heap allowance.
- Approximately 100KB is allocated for each compressed table that is accessed. If you are using table partitioning, this amount is per table partition. Up to 100MB is required for each column-based table that is accessed. These memory requirements are not included in the initial database heap sizing, and must be included in any fixed **dbheap** value. This metadata is cached while the database is active.
- If **AUDIT_BUF_SIZE** is set to a non-zero value, two buffers of the specified size are allocated during database activation. This is included in the internal database heap allowance.
- An allowance for two HADR buffers is made for the initial database heap reservation. For example, if **DB2_HADR_BUF_SIZE** is set to 256000 4KB pages, approximately 2GB is added to the initial database heap reservation from database memory. This is part of the internal database heap allowance, and occurs on both the primary and standby databases to support the resources required for failover operations.

Monitoring

The **dbheap** value that you configure represents only a portion of the database heap that is allocated. As documented, an extra amount is added to the database heap to account for critical requirements. Therefore, it is normal for database heap memory usage to exceed the user-configured value for the **dbheap** parameter.

You can monitor your database heap usage by using the **MON_GET_MEMORY_POOL** table function. For example, the following query:

```
select memory_pool_used, memory_pool_used_hwm
from table (mon_get_memory_pool(null,null,null))
where memory_pool_type='DATABASE'
```

Returns the values of **memory_pool_used** and **memory_pool_used_hwm** in KB units:

```
MEMORY_POOL_USED      MEMORY_POOL_USED_HWM
-----
140574                140574
```

1 record(s) selected.

You can also use the **db2pd -db <database_name> -mempools** command to monitor database heap usage.

db_mem_thresh - Database memory threshold

This parameter represents the maximum percentage of committed, but currently unused, database shared memory that the database manager will allow before starting to release committed pages of memory back to the operating system.

Configuration type

Database

Parameter type

- Configurable online

- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

100 [0-100]

Unit of measure

Percentage

This database configuration parameter relates to how the database manager handles excess unused database shared memory. Typically, as pages of memory are touched by a process, they are committed, meaning that a page of memory is allocated by the operating system and occupies space either in physical memory or in a page file on disk. Depending on the database workload, there might be peak database shared memory requirements at a certain times of day. Once the operating system has enough committed memory to meet those peak requirements, that memory remains committed, even after peak memory requirements have subsided.

A value of 0 means to immediately release any unused database shared memory, and a value of 100 means to never release any unused database shared memory. The default for new (not upgraded) databases is 100, which is suitable for most workloads.

This configuration parameter can be updated dynamically. Care should be taken when updating this parameter, as setting the value too low could cause excessive memory thrashing on the box (memory pages constantly being committed and then released), and setting the value too high might prevent the database manager from returning any database shared memory back to the operating system for other processes to use.

This configuration parameter is ignored (meaning that unused database shared memory pages remain committed) if the database shared memory region is pinned through the **DB2_PINNED_BP** registry variable, configured for large pages through the **DB2_LARGE_PAGE_MEM** registry variable, or if releasing of memory is explicitly disabled through the **DB2MEMDISCLAIM** registry variable.

Some versions of Linux do not support releasing subranges of a shared memory segment back to the operating system. On such platforms, this parameter is ignored.

date_compat - Date compatibility database configuration parameter

This parameter indicates whether the DATE compatibility semantics associated with the **TIMESTAMP(0)** data type are applied to the connected database.

Configuration type

Database

Parameter type

Informational

The value is determined at database creation time, and is based on the setting of the **DB2_COMPATIBILITY_VECTOR** registry variable for DATE data type. The value cannot be changed.

dec_arithmetic - DECIMAL arithmetic mode

This parameter specifies the DECIMAL arithmetic mode. This mode affects the rules for result precision and scale of basic DECIMAL operators (+, -, *, /) and the AVG and SUM aggregate functions with a DECIMAL argument.

Configuration type

Database

Parameter type

Configurable

Default [range]

NULL [DEC.S, DEC15, 15, DEC31, 31, D15.S, D31.S where S is a digit in the range 1 - 9]

The **dec_arithmetic** database configuration parameter can be used to change the resulting scale of a decimal arithmetic operation that involves division. Imposing a minimum division scale can be used to avoid SQLSTATE 42911.

This parameter can also be used to enable Db2 for z/OS DECIMAL arithmetic emulation. See “Expressions”, “AVG aggregate function”, and “SUM aggregate function” in *SQL Reference* for details on the effects of Db2 for z/OS emulation modes.

NULL

Specifies the default Db2 rules with no minimum DECIMAL division scale.

DEC.S or DEC,S

Specifies the default Db2 rules with a minimum DECIMAL division scale, S must be in the range 1 - 9. The separator that is used can be either a period or a comma, regardless of the setting of the default decimal point.

DEC15 or 15

Db2 for z/OS compatibility mode, equivalent to Db2 for z/OS **DECARTH DECP** value DEC15 or 15. Specifies the rules that do not allow a precision that is greater than 15 digits.

DEC31 or 31

Db2 for z/OS compatibility mode, equivalent to Db2 for z/OS **DECARTH DECP** value DEC31 or 31.

D15.S or D15,S

Db2 for z/OS DEC15 compatibility mode with a minimum DECIMAL division scale, S must be in the range 1 - 9. The separator that is used can be either a period or a comma, regardless of the setting of the default decimal point.

D31.S or D31,S

Db2 for z/OS DEC31 compatibility mode with a minimum DECIMAL division scale, S must be in the range 1 - 9. The separator that is used can be either a period or a comma, regardless of the setting of the default decimal point.

Effects of changing the value of dec_arithmetic

Changing this database configuration parameter might cause changes to applications for existing databases. This behavior can occur when the resulting scale for decimal division would be impacted by changing this database configuration parameter. The following list shows some possible scenarios that might affect applications. Consider these scenarios before you change the **dec_arithmetic** configuration parameter on a database server with existing databases:

- A static package does not change behavior until the package is rebound, either implicitly or explicitly. For example, after you change the value from DEC.3 to DEC.6, the additional scale digits might not be included in the results until rebind occurs. After you change the value of **dec_arithmetic**, recompile all static SQL packages whose results are effected by the change to force a rebind operation. You can force an explicit rebind by running the **REBIND** command or the **db2rbind** command.
- Materialized query tables (MQTs) might contain different results after you alter the **dec_arithmetic** configuration parameter. To ensure that previously created MQTs contain only data that adheres to the new format, refresh these MQTs by using the REFRESH TABLE statement.
- The results of a trigger might be affected by the changed format. Altering the **dec_arithmetic** value has no effect on data that is already written.
- A check constraint that involves decimal division might restrict some values that were previously accepted. Such rows now violate the constraint but are not detected until one of the following events occurs:
 - One of the columns that are involved in the check constraint row is updated.
 - The SET INTEGRITY statement with the IMMEDIATE CHECKED option is processed.

The following steps force the check of such constraints:

1. Run the ALTER TABLE statement to drop the check constraint.
 2. Run the ALTER TABLE statement to add the constraint.
- After you change the value of **dec_arithmetic**, recompile all static SQL packages that depend on the value of a generated column whose results are effected by the change in the **dec_arithmetic** value. You can rebind all the packages by running the **db2rbind** command.
 - An index with expression-based keys whose calculation depends on **dec_arithmetic** might be different for identical rows. The difference in values occurs if one row was inserted before the change to **dec_arithmetic** and the other was inserted after. Drop and re-create all potentially impacted expression-based indexes after you change the value of the **dec_arithmetic** configuration parameter. If you are unsure that a particular expression-based index is impacted, drop and re-create the index to avoid incorrect values in the index.

dec_to_char_fmt - Decimal to character function configuration parameter

This parameter is used to control the result of the CHAR scalar function and the CAST specification for converting decimal to character values.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

See “Effects of changing the value of dec_to_char_fmt” on page 148.

Default [range]

NEW [NEW, V95]

The setting of the parameter determines whether leading zeros and a trailing decimal characters are included in the result of the CHAR function. If you set the

parameter to `NEW`, leading zeros and a trailing decimal characters are not included; if you set the parameter to `V95`, leading zeros and a trailing decimal characters are included.

Leading zeros and a trailing decimal characters are also included in the result of the `CHAR_OLD` scalar function, which has the same syntax as the `CHAR` function.

When upgrading, for databases created before Version 9.7 and then upgraded to Version 9.7 or higher, the parameter `dec_to_char_fmt` is set to `V95` by default.

Effects of changing the value of `dec_to_char_fmt`

- Materialized query tables (MQTs) that you created before Version 9.7 might contain results that differ from those MQTs that you created by using the `NEW` setting. To ensure that previously created MQTs contain only data that adheres to the new format, refresh these MQTs by using the `REFRESH TABLE` statement.
- The results of a trigger might be affected by the changed format. Setting the value of the parameter to `NEW` to change the format has no effect on data that has already been written.
- Constraints that allowed data to be inserted into a table might, if reevaluated, reject that same data. Similarly, constraints that did not allow data to be inserted into a table might, if reevaluated, accept that same data. Use the `SET INTEGRITY` statement to check for and correct data in a table that might no longer satisfy a constraint.
- After changing the value of `dec_to_char_fmt`, recompile all static SQL packages that depend on the value of a generated column whose results are effected by the change in the `dec_to_char_fmt` value. To find out which static SQL packages are effected, you must compile, rebind all the packages using the `db2rbind` command.
- The value of an index with expression-based keys whose calculation is dependent on `dec_to_char_fmt` will be different after changing the value of `dec_to_char_fmt`. Drop and recreate all potentially impacted expression-based indexes after changing the value of `dec_to_char_fmt`. If you are not sure that a particular expression-based index is impacted, it is best to drop and recreate the index to avoid incorrect values in the index.

decflt_rounding - Decimal floating point rounding configuration parameter

This parameter specifies the rounding mode for decimal floating point (DECFLOAT) values. The rounding mode affects decimal floating-point operations in the server, and in `LOAD` command operations.

Configuration type

Database

Parameter type

Configurable

See “Effects of changing the value of `decflt_rounding`” on page 149.

Default [range]

`ROUND_HALF_EVEN` [`ROUND_CEILING`, `ROUND_FLOOR`, `ROUND_HALF_UP`, `ROUND_DOWN`]

Db2 database systems support five IEEE-compliant decimal floating point rounding modes. The rounding mode specifies how to round the result of a calculation when the result exceeds the precision. The definitions for all the rounding modes are as follows:

ROUND_CEILING

Round towards +infinity. If all of the discarded digits are zero or if the sign is negative the result is unchanged. Otherwise, the result coefficient is incremented by 1 (rounded up).

ROUND_FLOOR

Round towards -infinity. If all of the discarded digits are zero or if the sign is positive the result is unchanged. Otherwise, the sign is negative and the result coefficient should be incremented by 1.

ROUND_HALF_UP

Round to nearest digit or, if equidistant, round up by 1. If the discarded digits represent a value that is greater than or equal to 0.5 in the next left position, then the result coefficient is incremented by 1 (rounded up). Otherwise, the discarded digits that have a value that is less than 0.5 are ignored.

ROUND_HALF_EVEN

Round to nearest digit or, if equidistant, round so that the final digit is an even number. If the discarded digits represent a value greater than 0.5 in the next left position, then the resulting coefficient is increment by 1 (rounded up). If they represent less than half, then the result coefficient is not adjusted, that is, the discarded digits are ignored. Otherwise, if they represent exactly half, the result coefficient is unaltered if its rightmost digit is even, or incremented by 1 (rounded up) if its rightmost digit is odd, to make an even digit. This rounding mode is the default rounding mode as stated in the IEEE decimal floating point specification and is the default rounding mode in Db2 database products.

ROUND_DOWN

Round towards 0 (truncation). The discarded digits are ignored.

Table 12 shows the result of rounding of 12.341, 12.345, 12.349, 12.355, and -12.345, each to 4 digits, under different rounding modes:

Table 12. Decimal floating point rounding modes

Rounding mode	12.341	12.345	12.349	12.355	-12.345
ROUND_DOWN	12.34	12.34	12.34	12.35	-12.34
ROUND_HALF_UP	12.34	12.35	12.35	12.36	-12.35
ROUND_HALF_EVEN	12.34	12.34	12.35	12.36	-12.34
ROUND_FLOOR	12.34	12.34	12.34	12.35	-12.35
ROUND_CEILING	12.35	12.35	12.35	12.36	-12.34

Effects of changing the value of `decflt_rounding`

Changing the value of the parameter has the following consequences:

- Previously constructed materialized query tables (MQTs) could contain results that differ from what would be produced with the new rounding mode. To correct this problem, refresh potentially impacted MQTs.
- The results of a trigger might be affected by the new rounding mode. Changing it has no effect on data that has already been written.
- Constraints that allowed data to be inserted into a table, if reevaluated, might reject that same data. Similarly constraints that did not allow data to be inserted into a table, if reevaluated, might accept that same data. Use the SET INTEGRITY statement to check for and correct such problems.

- The value of a generated column or an index with expression-based keys whose calculation is dependent on **decflt_rounding** might be different for identical rows. The difference in values occurs if one row was inserted before the change to **decflt_rounding** and the other was inserted after. Drop and recreate all potentially impacted expression-based indexes after changing the value of the **decflt_rounding** configuration parameter. If you are not sure that a particular expression-based index is impacted, it is best to drop and recreate the index to avoid incorrect values in the index.
- The value of **decflt_rounding** is not compiled into sections. Therefore, you do not have to recompile static SQL statements after changing the value of the **decflt_rounding** configuration parameter.

The value of this configuration parameter is not changed dynamically. The changes become effective only after all applications disconnect from the database and the database is restarted.

dft_degree - Default degree

This parameter specifies the default value for the CURRENT DEGREE special register and the **DEGREE** bind option.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Connection

Default [range]

1 [-1(ANY), 1 - 32 767]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

The default value is 1.

A value of 1 means no intrapartition parallelism. A value of -1 (or ANY) means the optimizer determines the degree of intrapartition parallelism based on the number of processors and the type of query.

The degree of intrapartition parallelism for an SQL statement is specified at statement compilation time using the CURRENT DEGREE special register or the **DEGREE** bind option. The maximum runtime degree of intrapartition parallelism for an active application is specified using the **SET RUNTIME DEGREE** command. The Maximum Query Degree of Parallelism (**max_querydegree**) configuration parameter specifies the maximum query degree of intrapartition parallelism for all SQL queries.

The actual runtime degree used is the lowest of:

- **max_querydegree** configuration parameter
- application runtime degree
- SQL statement compilation degree

dft_extent_sz - Default extent size of table spaces

This parameter sets the default extent size of table spaces.

Configuration type

Database

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

32 [2 - 256]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages

When a table space is created, EXTENTSIZE n can be optionally specified, where n is the extent size. If you do not specify the extent size on the CREATE TABLESPACE statement, the database manager uses the value given by this parameter.

Recommendation: In many cases, you will want to explicitly specify the extent size when you create the table space. Before choosing a value for this parameter, you should understand how you would explicitly choose an extent size for the CREATE TABLESPACE statement.

In a Db2 pureScale environment, you should use an extent size of at least the default (32 pages). This minimum extent size reduces the amount of internal message traffic within the Db2 pureScale environment when extents are added for a table or index. Examples of cases where new extents are allocated frequently, and where a larger extent size is beneficial, include the load and import utilities, the CREATE INDEX statement, and bulk insert operations from applications.

dft_loadrec_ses - Default number of load recovery sessions

This parameter specifies the default number of sessions that will be used during the recovery of a table load.

Configuration type

Database

Parameter type

- Configurable online

Propagation class

Immediate

Default [range]

1 [1 - 30 000]

Unit of measure

Counter

The value of this parameter should be set to the number of I/O sessions that was specified with the COPY YES option in the original **LOAD** command. The retrieval of a

load copy is an operation similar to restore. You can override this parameter through entries in the copy location file specified by the environment variable **DB2LOADREC**.

The default number of buffers used for load retrieval is two more than the value of this parameter. You can also override the number of buffers in the copy location file.

This parameter is applicable only if roll forward recovery is enabled.

dft_mttb_types - Default maintained table types for optimization

This parameter specifies the default value for the CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register. The value of this register determines what types of refresh deferred materialized query tables will be used during query optimization.

Configuration type

Database

Parameter type

Configurable

Default [range]

SYSTEM [ALL, NONE, FEDERATED_TOOL, SYSTEM, USER, REPLICATION, or a list of values]

You can specify a list of values that are separated by commas, for example, 'USER,FEDERATED_TOOL'. ALL or NONE cannot be listed with other values, and you cannot specify the same value more than once.

When you set the **dft_mttb_types** configuration parameter, if the value includes anything other than REPLICATION or NONE, then the **dft_refresh_age** must have a value of either 0 or 99999999999999 (ANY).

For use with the **db2CfgSet** and **db2CfgGet** APIs, the acceptable parameter values are: 8 (ALL), 4 (NONE), 16 (FEDERATED_TOOL), 32 (REPLICATION), 1 (SYSTEM) and 2 (USER). Multiple values can be specified together by using bitwise OR; for example, 18 would be the equivalent of USER,FEDERATED_TOOL. As before, the values 4 and 8 cannot be used with other values.

dft_prefetch_sz - Default prefetch size

This parameter sets the default prefetch size of table spaces.

Configuration type

Database

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Automatic [0 - 32 767]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages

When a table space is created, PREFETCHSIZE can optionally be specified with a value of AUTOMATIC or *n*, where *n* represents the number of pages the database manager will read if prefetching is being performed. If you do not specify the prefetch size on invocation of the CREATE TABLESPACE statement, the database manager uses the current value of the **dft_prefetch_sz** parameter.

If a table space is created with the prefetch size set to AUTOMATIC, the Db2 database manager will automatically calculate and update the prefetch size of the table space.

This calculation is performed:

- When the database starts
- When a table space is first created with AUTOMATIC prefetch size
- When the number of containers for a table space changes through execution of an ALTER TABLESPACE statement
- When the prefetch size for a table space is updated to be AUTOMATIC through execution of an ALTER TABLESPACE statement

The AUTOMATIC state of the prefetch size can be turned on or off as soon as the prefetch size is updated manually through invocation of the ALTER TABLESPACE statement.

Recommendation: Using system monitoring tools, you can determine if your CPU is idle while the system is waiting for I/O. Increasing the value of this parameter can help if the table spaces being used do not have a prefetch size defined for them.

This parameter provides the default for the entire database, and it might not be suitable for all table spaces within the database. For example, a value of 32 might be suitable for a table space with an extent size of 32 pages, but not suitable for a table space with an extent size of 25 pages. Ideally, you should explicitly set the prefetch size for each table space.

To help minimize I/O for table spaces defined with the default extent size (**dft_extent_sz**), you should set this parameter as a factor or whole multiple of the value of the **dft_extent_sz** parameter. For example, if the **dft_extent_sz** parameter is 32, you could set **dft_prefetch_sz** to 16 (a fraction of 32) or to 64 (a whole multiple of 32). If the prefetch size is a multiple of the extent size, the database manager might perform I/O in parallel, if the following conditions are true:

- The extents being prefetched are on different physical devices
- Multiple I/O servers are configured (**num_ioservers**).

dft_queryopt - Default query optimization class

The query optimization class is used to direct the optimizer to use different degrees of optimization when compiling SQL and XQuery queries.

This parameter provides additional flexibility by setting the default query optimization class used when neither the SET CURRENT QUERY OPTIMIZATION statement nor the **QUERYOPT** option on the **BIND** command are used.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Connection

Default [range]

5 [0 - 9]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Query Optimization Class (see the following list)

The query optimization classes currently defined are:

- 0 - minimal query optimization.
- 1 - roughly comparable to Db2 Version 1.
- 2 - slight optimization.
- 3 - moderate query optimization.
- 5 - significant query optimization with heuristics to limit the effort expended on selecting an access plan. This is the default.
- 7 - significant query optimization.
- 9 - maximal query optimization

dft_refresh_age - Default refresh age

This parameter specifies the default value for the CURRENT REFRESH AGE special register. The value represents the maximum duration since a particular time-stamped event occurred to a cached data object. The cached data object can be used during this period to help optimize the processing of a query. An example of a time-stamped event is processing a REFRESH TABLE statement on a system-maintained REFRESH DEFERRED materialized query table.

Configuration type

Database

Parameter type

Configurable

Default [range]

0 [0 - 99999999999999 (ANY)]

Unit of measure

None

This parameter specifies a time stamp value with a data type of DECIMAL(20,6). The value must be 0 - 99999999999999 or a valid time stamp within that range. The valid format for the range is *yyyymmddhhmmss.nnnnnn*, where:

- *yyyy* is the number of years and can have a value of 0 - 9999.
- *mm* is the number of months and can have a value of 0 - 11.
- *dd* is the number of days and can have a value of 0 - 30.
- *hh* is the number of hours and can have a value of 0 - 23.

- *mm* is the number of minutes and can have a value of 0 - 59.
- *ss* is the number of seconds and can have a value of 0 - 59.
- *nnnnnn* is the number of fractional seconds. The fractional seconds portion of the value is ignored and therefore can be any value.

You do not have to include the leading zeros for the entire value or the trailing fractional seconds. However, individual elements that have another element to the left must include the zeros. For example, to represent 1 hour, 7 minutes, and 5 seconds, use 10705.

To set the **dft_refresh_age** configuration parameter to a value other than 0 or 9999999999999999 (ANY), you must set the **dft_mttb_types** database configuration parameter to REPLICATION or NONE.

If the CURRENT REFRESH AGE has a value of 9999999999999999 (ANY), and the QUERY OPTIMIZATION class has a value of two, five or more, REFRESH DEFERRED materialized query tables are considered to optimize the processing of a dynamic query.

dft_schemas_dcc - Default data capture on new schemas configuration parameter

This parameter allows the control of default setting for DATA CAPTURE CHANGES on newly created schemas for replication purposes.

Configuration type

Database

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

No [Yes; No]

By default, **dft_schemas_dcc** is set to NO. When set to YES, all newly created schemas by default will have the DATA CAPTURE CHANGES clause.

dft_sqlmathwarn - Continue upon arithmetic exceptions

This parameter sets the value that determines the handling of arithmetic errors, such as division by zero, and retrieval conversion errors during SQL statement execution.

Configuration type

Database

Parameter type

Configurable

Default [range]

No [No, Yes]

For static SQL statements, the value of this parameter is associated with the package at bind time. For dynamic SQL data manipulation language (DML) statements, the value of this parameter is used when the statement is prepared.

Recommendation: Use the default setting of No, unless you specifically require queries to be processed that include arithmetic exceptions. Then specify the value of Yes. The processing of queries that include arithmetic exceptions can occur if you are processing SQL statements that, on other database managers, provide results regardless of the arithmetic exceptions that occur.

Restriction: A value of Yes is not supported when column-organized tables are accessed.

Effects of changing the value of `dft_sqlmathwarn`

If you change the `dft_sqlmathwarn` value for a database, the behavior of check constraints, triggers, views, and indexes with expression-based keys that include arithmetic expressions might change. This might, in turn, have an impact on the data integrity of the database. Only change the setting of the `dft_sqlmathwarn` configuration parameter for a database after you have carefully evaluated how the new arithmetic exception handling behavior might impact check constraints, triggers, views, and indexes with expression-based keys. Drop and recreate all potentially impacted expression-based indexes after changing the value of the `dft_sqlmathwarn` configuration parameter. If you are not sure that a particular expression-based index is impacted, it is best to drop and recreate the index to avoid incorrect values in the index.

Consider the following check constraint, which includes a division arithmetic operation:

```
A/B > 0
```

When `dft_sqlmathwarn` is No and an INSERT statement with B=0 is attempted, the division by zero is processed as an arithmetic error. The insert operation fails because the Db2 database manager cannot check the constraint. If `dft_sqlmathwarn` is changed to Yes, the division by zero is processed as an arithmetic warning with a NULL result. The NULL result causes the predicate to evaluate to UNKNOWN and the insert operation succeeds. If `dft_sqlmathwarn` is changed back to No, an attempt to insert the same row will fail, because the division by zero error prevents the Db2 database manager from evaluating the constraint. The row that was inserted with B=0 when `dft_sqlmathwarn` was set to Yes remains in the table and can be selected. Updates to the row that cause the constraint to be evaluated fail, and updates to the row that do not require constraint reevaluation will succeed.

Before changing `dft_sqlmathwarn` from No to Yes, you should consider rewriting the constraint to explicitly handle nulls from arithmetic expressions. For example, the following code can be used if both A and B are nullable.:

```
( A/B > 0 ) AND ( CASE
    WHEN A IS NULL THEN 1
    WHEN B IS NULL THEN 1
    WHEN A/B IS NULL THEN 0
    ELSE 1
    END
= 1 )
```

And, if A or B is not-nullable, the corresponding IS NULL WHEN-clause can be removed.

Before changing `dft_sqlmathwarn` from Yes to No, you must first check for data that might become inconsistent by using predicates such as the following:

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

If you isolate inconsistent rows, you must take the appropriate actions to correct the inconsistency before changing **dft_sqlmathwarn**. You can also manually recheck constraints with arithmetic expressions after the change. To manually check constraints, place the affected tables in a check pending state with the OFF clause of the SET CONSTRAINTS statement, and request that the tables be checked with the IMMEDIATE CHECKED clause of the SET CONSTRAINTS statement. Inconsistent data is indicated by an arithmetic error, which prevents the constraint from being evaluated.

dft_table_org - Default table organization

This parameter specifies whether a user table is created as a column-organized table or a row-organized table if you do not specify the ORGANIZE BY COLUMN or the ORGANIZE BY ROW clause for the CREATE TABLE statement.

Configuration type

Database

Parameter type

Configurable online

Default [range]

ROW [COLUMN, ROW]

COLUMN Specifies that regular user tables are created as column-organized tables unless you specify the ORGANIZE BY ROW clause for the CREATE TABLE statement. A default setting of COLUMN is ignored for the following types of tables:

- Materialized query table
- Range clustered table
- Multidimensional clustering table
- Insert time clustering table
- Global temporary table
- Typed table

You can create a materialized query table as a column-organized MQT, but the ORGANIZE BY COLUMN clause must be specified when creating the MQT, even when **dft_table_org** is set to COLUMN.

ROW Specifies that regular user tables are created as row-organized tables unless you specify the ORGANIZE BY COLUMN clause for the CREATE TABLE statement. The ROW option is the default setting, unless the default table organization was set to COLUMN automatically by setting the **DB2_WORKLOAD** registry variable to ANALYTICS *prior* to creating the database.

If you set the **DB2_WORKLOAD** registry variable to ANALYTICS and you do not disable the AUTOCONFIGURE option, a newly created database or a database that you configure by using the Configuration Advisor uses a value of COLUMN for the **dft_table_org** configuration parameter. Note that a newly created database will be auto-configured by default unless the **DB2_ENABLE_AUTOCONFIG_DEFAULT** registry variable is set to NO, or the AUTOCONFIGURE APPLY NONE clause is specified on the **CREATE DATABASE** command.

discover_db - Discover database

You can use this parameter to prevent information about a database from being returned to a client when a discovery request is received at the server.

Configuration type

Database

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

ENABLE [DISABLE, ENABLE]

By changing this parameter value to `DISABLE`, it is possible to hide databases with sensitive data from the discovery process. This can be done in addition to other database security controls on the database.

dlchktime - Time interval for checking deadlock

This parameter defines the frequency at which the database manager checks for deadlocks among all the applications connected to a database.

Configuration type

Database

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

10 000 (10 seconds) [1 000 - 600 000]

Unit of measure

Milliseconds

A deadlock occurs when two or more applications connected to the same database wait indefinitely for a resource. The waiting is never resolved because each application is holding a resource that the other needs to continue.

Note:

1. In a partitioned database environment, this parameter applies to the catalog node only.
2. In a partitioned database environment, a deadlock is not flagged until after the second iteration.

Recommendation: Increasing this parameter decreases the frequency of checking for deadlocks, thereby increasing the time that application programs must wait for the deadlock to be resolved.

Decreasing this parameter increases the frequency of checking for deadlocks, thereby decreasing the time that application programs must wait for the deadlock to be resolved but increasing the time that the database manager takes to check for deadlocks. If the deadlock interval is too small, it can decrease runtime performance, because the database manager is frequently performing deadlock detection. If this parameter is set lower to improve concurrency, you should ensure that `maxlocks` and `locklist` are set appropriately to avoid unnecessary lock escalation, which can result in more lock contention and as a result, more deadlock situations.

enable_xmlchar - Enable conversion to XML configuration parameter

This parameter determines whether XMLPARSE operations can be performed on non-BIT DATA CHAR (or CHAR-type) expressions in an SQL statement.

Configuration type

Database

Parameter type

Configurable

Default [range]

Yes [Yes; No]

When pureXML[®] features are used in a non-Unicode database, the XMLPARSE function can cause character substitutions to occur as SQL string data is converted from the client code page into the database code page, and then into Unicode for internal storage. Setting **enable_xmlchar** to NO blocks the usage of character data types during XML parsing, and any attempts to insert character types into a non-Unicode database will generate an error. The BLOB data type and FOR BIT DATA data types are still allowed when **enable_xmlchar** is set to NO, as code page conversion does not occur when these data types are used to pass XML data into a database.

By default, **enable_xmlchar** is set to YES so that parsing of character data types is allowed. In this case, you should ensure that any XML data to be inserted contains only code points that are part of the database code page, in order to avoid substitution characters being introduced during insertion of the XML data.

Note: The client needs to disconnect and reconnect to the agent for this change to be reflected.

encrib - Encryption library

The **encrib** configuration parameter enables automatic encryption of backups. Use the **encrib** parameter to specify the full absolute path to the encryption library which plugs in to the Db2 compression API, such as IBM InfoSphere[®] Guardium[®] Data Encryption.

Configuration type

Database

Parameter type

Configurable online (requires a database connection)

Propagation class

Immediate

Default [range]

NULL [*<path-to-encryption-library>*]

By default, the **encrib** configuration parameter is set to NULL, meaning that backups are not automatically encrypted. To have your backups encrypted, either specify the **ENCRYPT** option with the **BACKUP DATABASE** command (to have that specific backup encrypted), or have **encrib** set to a non-NULL value (to have all backups automatically encrypted). This enforced encryption does not apply to snapshot backups, which are not encrypted. When the **encrib** configuration parameter is set, you cannot specify any compression options with your backup operations, and the only valid encryption option that you can specify is **EXCLUDE**.

Only a user with SECADM authorization can change the setting of the **enclib** configuration parameter.

The path used for the library should be a full absolute path. Relative paths are interpreted relative to the current working directory of the Db2 server. The path is interpreted while it is being set to resolve all symbolic links and relative path references such as `...`. This fully-expanded path is stored in the database configuration.

encropts - Encryption options

The **encropts** configuration parameter specifies a string of options for automatic encryption of backups. Use the **encropt** parameter in tandem with the **enclib** parameter.

Configuration type

Database

Parameter type

Configurable online (requires a database connection)

Propagation class

Immediate

Default [range]

NULL [*<string>*]

By default, the **encropts** configuration parameter is set to NULL. Any non-NULL settings are ignored if automatic backup encryption is not enabled (if **enclib** has a NULL setting).

The format of the **encropts** string is described under the **COMPROPTS|ENCROPTS** keyword of the BACKUP DATABASE command.

Only a user with SECADM authorization can change the setting of the **encropts** configuration parameter.

extended_row_sz - Extended row size

You can use the **extended_row_sz** configuration parameter to control if a table definition can exceed the maximum row length of a page.

Configuration type

Database

Parameter type

Configurable online

Default [range]

ENABLE [DISABLE, ENABLE]

If you are upgrading from Version 10.1 or earlier, the default value of **extended_row_sz** is DISABLE.

When **extended_row_sz** is set to DISABLE, extended row size is disabled.

When **extended_row_sz** is set to ENABLE, extended row size is enabled. During an INSERT or UPDATE table operation, row data that exceeds the maximum row length moves a subset of VARCHAR, VARBINARY, or VARGRAPHIC data out of

the row and stores it as LOB data. While the data is in LOB format, data access might be slower because additional I/O resources might be required to fetch, insert, or update the LOB data.

failarchpath - Failover log archive path

This parameter specifies a path to which the Db2 database system will try to archive log files if the log files cannot be archived to either the primary or the secondary (if set) archive destinations because of a media problem affecting those destinations. This specified path must reference a disk.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Default [range]

Null []

If there are log files in the path specified by the current value of **failarchpath**, any updates to **failarchpath** will not take effect immediately. Instead, the update will take effect when all applications disconnect.

groupheap_ratio - Percent of memory for application group heap

This parameter has been deprecated since Version 9.5, but is still being used by pre-Version 9.5 data servers and clients. Any value specified for this configuration parameter will be ignored by the database manager in Db2 Version 9.5 or later releases.

In Version 9.5, it has been replaced by the **app1_memory** configuration parameter.

Note: The following information applies only to pre-Version 9.5 data servers and clients.

This parameter specifies the percentage of memory in the application control shared memory set devoted to the application group shared heap.

Configuration type

Database

Parameter type

Configurable

Default [range]

70 [1 - 99]

Unit of measure

Percentage

This parameter does not have any effect on a non-partitioned database with concentrator OFF and intrapartition parallelism disabled.

Recommendation: Retain the default value of this parameter unless you are experiencing performance problems.

hadr_db_role - HADR database role

This parameter indicates the current role of a database, whether the database is online or offline.

Configuration type
Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type
Informational

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_db_role** is set to the value on member 0.

Valid values are: STANDARD, PRIMARY, or STANDBY.

Note: When a database is active, the HADR role of the database can also be determined using the **GET SNAPSHOT FOR DATABASE** command.

hadr_local_host - HADR local host name

This parameter specifies the local host for high availability disaster recovery (HADR) TCP communication.

Configuration type
Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

- Configurable⁵
- Configurable by member in a Db2 pureScale environment

Default
Null

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_local_host** is set to the value on member 0.

Either a host name or an IP address can be used. If a host name is specified and it maps to multiple IP addresses, an error is returned, and HADR will not start up. If the host name maps to multiple IP addresses (even if you specify the same host

5. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

name on primary and standby), primary and standby can end up mapping this host name to different IP addresses, because some DNS servers return IP address lists in non-deterministic order.

A host name is in the form: myserver.ibm.com. An IP address is in the form: "12.34.56.78".

Usage Notes

- If your primary and standby start but do not connect to one another, and no error indication appears in the Db2 diagnostic log, it could be due to subtle issues in host name resolution. If you configured HADR using host names, try again using the IP addresses instead.
- If either of the primary or standby resides in a private network behind a NAT (Network Address Translation) device, you might need to set the registry variable **DB2_HADR_NO_IP_CHECK** to 0N. Refer to HADR and Network Address Translation (NAT) support

hadr_local_svc - HADR local service name

This parameter specifies the TCP service name or port number for which the local high availability disaster recovery (HADR) process accepts connections.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

- Configurable⁶
- Configurable by member in a Db2 pureScale environment.

Default

Null

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_local_svc** is set to the value on member 0.

The value for **hadr_local_svc** on the primary or standby database systems cannot be the same as the value of **svcname** or **svcname** +1 on their corresponding hosts.

If you are using SSL, do not set **hadr_local_svc** on the primary or standby database system to the same value as you set for **ssl_svcname**.

hadr_peer_window - HADR peer window configuration parameter

When you set **hadr_peer_window** to a non-zero time value, then a HADR primary-standby database pair continues to behave as though still in peer state, for the configured amount of time, if the primary database loses connection with the standby database. This helps ensure data consistency.

6. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

Configuration type

Database

Parameter type

- Configurable⁷

Default [range]

0 [0 – 4 294 967 295]

Unit of measure

Seconds

Upgrade Note

- If you are upgrading from a Version 9.8 Fix Pack 4 Db2 pureScale environment or earlier, the value of **hadr_peer_window** is set to the value on member 0.

If you have not configured the **hadr_target_list** configuration parameter, the value for **hadr_peer_window** needs to be the same on both primary and standby databases. When **hadr_target_list** is set, the first standby listed (the *principal standby*) uses the primary's setting for **hadr_peer_window** and any setting for **hadr_peer_window** on the principal or auxiliary standbys is ignored unless one of them becomes the primary.

A recommended minimum value is 120 seconds.

The peer window is not currently supported in a Db2 pureScale environment, so **hadr_peer_window** value must be set to 0 for a Db2 pureScale instance. Attempts to set **hadr_peer_window** to a non-zero value in a Db2 pureScale instance fail with a warning, and the **START HADR** command fails if **hadr_peer_window** is already a non-zero value in a Db2 pureScale instance. When the **hadr_syncmode** value is set to **ASYNC** or **SUPERASYNC**, the **hadr_peer_window** value must be set to 0. Attempts to make configuration updates that violate this condition fail with a warning.

To avoid impacting the availability of the primary database when the standby database is intentionally shut down, for example, for maintenance, the peer window is not invoked if the standby database is explicitly deactivated while the HADR pair is in peer state.

The **TAKEOVER HADR** command with the **PEER WINDOW ONLY** option launches a takeover operation only if the HADR standby is presently inside the defined peer window.

The takeover operation with the **hadr_peer_window** parameter might behave incorrectly if the primary database clock and the standby database clock are not synchronized to within 5 seconds of each other. That is, the operation might succeed when it should fail, or fail when it should succeed. You should use a time synchronization service (for example, NTP) to keep the clocks synchronized to the same source.

On the standby database, the peer window end time is a time specified in the last heartbeat message that the standby received from the primary database, and is not directly related to when the standby detects loss of the connection.

7. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

hadr_remote_host - HADR remote host name

This parameter specifies the TCP/IP host name or IP address of the remote high availability disaster recovery (HADR) database server.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

- Configurable⁸

Default

Null

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_remote_host** is set to the value on member 0.

The information that is referenced by the **hadr_remote_host** configuration parameter is dependant on what role the server or cluster is assigned.

- On a primary, this configuration parameter references the host of the principal standby. In a Db2 pureScale environment, it contains a pipe-delimited (|) list of all addresses (hosts and ports) in the standby cluster.
- On a standby, this configuration parameter references the host of the primary. In a Db2 pureScale environment, it contains a pipe-delimited (|) list of all addresses (hosts and ports) in the primary cluster.

Unless you are using the Db2 pureScale feature, this parameter must have a non-NULL value in order to set up HADR. This parameter is under automatic configuration when the **hadr_target_list** configuration parameter is set, but you should still ensure that it has the correct value in case the integrated cluster manager, IBM Tivoli System Automation for Multiplatforms (SA MP), picks up the incorrect value before it is automatically reconfigured.

Similar to **hadr_local_host**, this parameter must map to only one IP address.

hadr_remote_inst - HADR instance name of the remote server

This parameter specifies the instance name of the remote server. High availability disaster recovery (HADR) also checks whether a remote database requesting a connection belongs to the declared remote instance.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

8. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

Parameter type
Configurable⁹

Default
Null

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_remote_inst** is set to the value on member 0.

The information that is referenced by the **hadr_remote_inst** configuration parameter is dependant on what role the server or cluster is assigned.

- On a primary, this configuration parameter references the instance name of the principal standby.
- On a standby, this configuration parameter references the instance name of the primary.

Unless you are using the Db2 pureScale feature, this parameter must have a non-NULL value in order to set up HADR. This parameter is under automatic configuration when the **hadr_target_list** configuration parameter is set, but you should still ensure that it has the correct value in case the integrated cluster manager, IBM Tivoli System Automation for Multiplatforms (SA MP), picks up the incorrect value before it is automatically reconfigured.

hadr_remote_svc - HADR remote service name

This parameter specifies the TCP service name or port number that will be used by the remote high availability disaster recovery (HADR) database server.

Configuration type
Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type
• Configurable¹⁰

Default
Null

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_remote_svc** is set to the value on member 0.

The information that is referenced by the **hadr_remote_svc** configuration parameter is dependant on the role that the server or cluster is assigned.

- On a primary, this configuration parameter references the port of the principal standby.

9. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

10. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

- On a standby, this configuration parameter references the port of the primary.

In a Db2 pureScale environment, this parameter is always set to NULL.

Unless you are using the Db2 pureScale feature, this parameter must have a non-NULL value in order to set up HADR. This parameter is under automatic configuration when the **hadr_target_list** configuration parameter is set, but you should still ensure that it has the correct value in case the integrated cluster manager, IBM Tivoli System Automation for Multiplatforms (SA MP), picks up the incorrect value before it is automatically reconfigured.

hadr_replay_delay - HADR replay delay configuration parameter

This parameter specifies the number of seconds that must pass from the time that a transaction is committed on the primary database to the time that the transaction is committed on the standby database.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

Configurable

Default [Range]

0 [0 to 2147483647]

Unit of measure

Seconds

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_replay_delay** is set to the value on member 0.

The **hadr_replay_delay** configuration parameter enables *delayed replay* on the HADR standby database, meaning that the standby intentionally lags the HADR primary database as the logs from the primary are replayed. The standby database must be in SUPERASYNC mode before **hadr_replay_delay** can be set to a nonzero value. You cannot set the parameter on the primary database. You must disable delayed replay on the standby before it can take over as the primary.

If you enable delayed replay, it is recommended that you also enable log spooling by setting the **hadr_spool_limit** database configuration parameter. Because of the intentional delay, replay position can be far behind log receive position on standby. Without spooling, log receive can only go beyond replay by the amount of the receive buffer. With spooling, the standby can receive much more logs beyond replay position, providing more protection against data loss in case of primary failure. Note that in either case, because of the mandatory SUPERASYNC mode, the primary will not be blocked by the delayed replay.

With delayed replay, if there is an errant transaction on the primary and it is noticed before it is replayed on the standby, you can roll forward the database to a

time that is right before when the errant transaction is committed, then stop roll forward to retrieve data lost on the primary.

Note: If you have delayed replay enabled on a standby, that standby cannot take over as the new primary until you have disabled delayed replay (set the `hadr_replay_delay` parameter to 0 on that standby).

hadr_spool_limit - HADR log spool limit configuration parameter

This parameter determines the maximum amount of log data that is allowed to be spooled to disk on HADR standby.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

Configurable¹¹

Default [range]

AUTOMATIC [-1 - 2 147 483 647]

Unit of measure

Pages (4 KB)

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of `hadr_spool_limit` is set to the value on member 0.

The `hadr_spool_limit` configuration parameter enables log spooling on the HADR standby database. Log spooling allows transactions on the HADR primary to make progress without having to wait for the log replay on HADR standby. Log data that is sent by the primary is then written, or *spooled*, to disk on the standby if it falls behind in log replay. The standby can later on read the log data from disk. This allows the system to better tolerate either a spike in transaction volume on the primary, or a slow down of log replay (due to the replay of particular type of log records) on the standby.

To disable spooling, set `hadr_spool_limit` to 0. When spooling is disabled, the standby can be behind the primary up to the size up the log receive buffer. When the buffer is full, it is possible that new transactions on the primary will be blocked because the primary cannot send any more log data to the standby system.

A value of -1 means unlimited spooling (as much as supported by the disk space available). If you are using a high value for `hadr_spool_limit`, you should consider that if there is a large gap between the log position of the primary and log replay

11. This configuration is only used by HADR standby database. The current standby database must be deactivated and activated again, in order for a change to take effect. This configuration can be changed online on the current primary database. The new value on the current primary database takes effect when the primary database changes its role to standby as a result of issuing the **TAKEOVER HADR** command.

on the standby, which might lead to a longer takeover time because the standby cannot assume the role of the new standby until the replay of the spooled logs finishes.

When making use of log spooling, ensure that adequate disk space is provided to the active log path of the standby database. There must be enough disk space to hold the active logs, which is determined by the **logprimary**, **logsecond**, **logfilesiz**, and **hadr_spool_limit** configuration parameters. To determine the current computed spool size (in pages), check the STANDBY_SPOOL_LIMIT field for the MON_GET_HADR table function or the **db2pd** command with the **-hadr** option.

Note that making use of log spooling does not compromise the HADR protection provided by the HADR feature. Data from the primary is still replicated in log form to the standby using the specified sync mode; it just takes time to apply (through log replay) the data to the table spaces.

HADR_SSL_LABEL - Label name in the key file for SSL communication between HADR primary and standby instances configuration parameter

This configuration parameter specifies the label of the SSL certificate which encrypts communication between primary and standby HADR instances in the key database.

Configuration type

Database

Parameter type

Configurable online¹²

Default [range]

NULL [certificate-label-name]

Specifies the label for the SSL certificate used to encrypt communication between primary and secondary HADR instances:

```
db2 update database configuration for <database_name> using HADR_SSL_LABEL <label_name>
```

HADR_SSL_LABEL accepts a label name of up to 127 characters. Specifying the **HADR_SSL_LABEL** parameter indicates that you would like SSL communication between primary and standby HADR instances. If an existing certificate expires and you specify a new label, existing encrypted connections will use the old certificate but any new connections created after will use the new certificate.

HADR_SSL_LABEL is currently supported on environments that do not use IBM Db2 pureScale. If you upgrade from Enterprise Server Edition (ESE) to Db2 pureScale while the **HADR_SSL_LABEL** is set, **db2checkSD** will return the error DBT5038N. Users should set the value to NULL before trying to upgrade to Db2 pureScale.

hadr_syncmode - HADR synchronization mode for log writes in peer state

This parameter specifies the synchronization mode, which determines how log writes on the primary are synchronized with log writes on the standby when the systems are in peer state.

12. Changes to this parameter does not affect HADR connection already established. Change takes effect on new connection between primary and standby.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

- Configurable¹³

Default [range]

Db2 pureScale environments

ASYNC [SYNC, NEARSYNC, ASYNC, SUPERASYNC]

Other environments

NEARSYNC [SYNC, NEARSYNC, ASYNC, SUPERASYNC]

Upgrade Note

- If you are upgrading from Db2 Version 9.8 Fix Pack 4 or earlier, the value of **hadr_syncmode** is set to the value on member 0.

Valid values for this parameter are as follows:

SYNC This mode provides the greatest protection against transaction loss, but at a cost of higher transaction response time.

In this mode, log writes are considered successful only when both of the following conditions are met:

- The log records were written to the log files on the primary database.
- The primary database received acknowledgement from the standby database that the log records were also written to log files on the standby database.

The log data is guaranteed to be stored at both sites.

NEARSYNC

This mode provides somewhat less protection against transaction loss, in exchange for a shorter transaction response time than that of SYNC mode.

In this mode, log writes are considered successful only when both of the following conditions are met:

- The log records were written to the log files on the primary database.
- The primary database received acknowledgement from the standby database that the log records were also written to main memory on the standby system.

Loss of data occurs only if both sites fail simultaneously and if the standby site has not transferred to nonvolatile storage all the log data that it received.

ASYNC

Compared with the SYNC and NEARSYNC modes, the ASYNC mode results in shorter transaction response times but might cause greater transaction losses if the primary database fails.

In this mode, log writes are considered successful only when both of the following conditions are met:

13. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

- The log records were written to the log files on the primary database.
- The log records were delivered to the TCP layer of the primary system's host.

Because the primary system does not wait for acknowledgement from the standby system, transactions might be considered committed when they are still on their way to the standby.

Note: The `hadr_syncmode` configuration parameter cannot be set to ASYNC if peer window functionality is enabled (that is, if `hadr_peer_window` has a nonzero value).

SUPERASYNC

This mode has the shortest transaction response time but also has the highest probability of transaction losses if the primary system fails. This mode is useful when you do not want transactions to be blocked or experience longer response times due to network interruptions or congestion.

In this mode, the HADR pair can never be in peer state or disconnected peer state. The log writes are considered successful only when the log records were written to the log files on the primary database. Because the primary system does not wait for acknowledgement from the standby system, transactions might be considered committed when they are still on their way to the standby.

Note: The `hadr_syncmode` configuration parameter cannot be set to SUPERASYNC if peer window functionality is enabled (that is, if `hadr_peer_window` has a nonzero value).

Figure 2 shows when the logs for transactions are considered successful based on the level of synchronization that you chose:

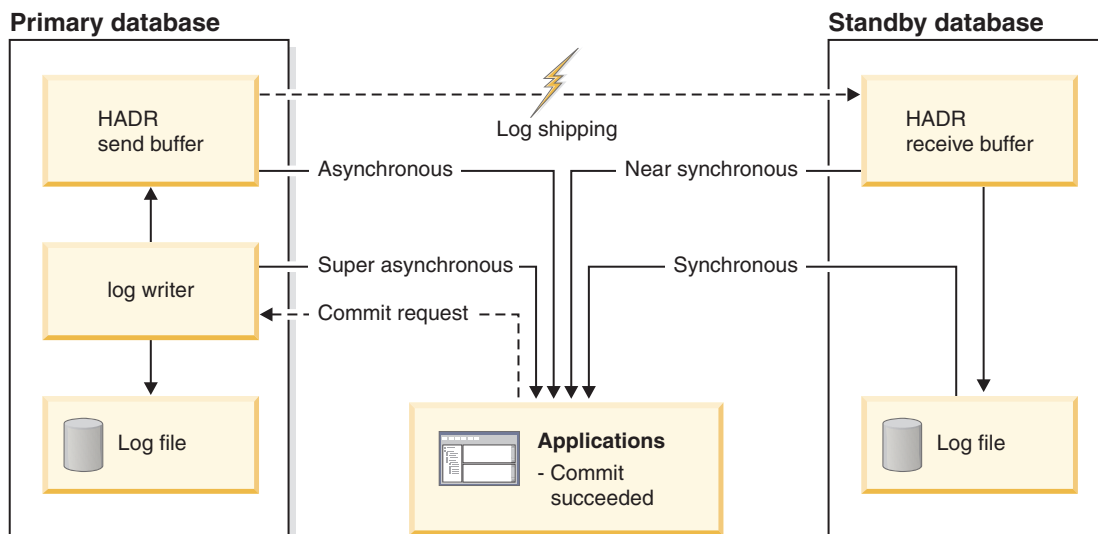


Figure 2. Synchronization modes for high availability disaster recovery (HADR)

Usage notes

Although you set the `hadr_syncmode` parameter on the primary and the standby databases, the effective synchronization mode is determined by the primary or by the standby's role. That is, auxiliary standbys (any standby that is not listed as the first entry in the primary's target list)

automatically have their synchronization modes set to SUPERASYNC. Also, the principal standby (the standby that is listed as the first entry in the primary's target list) uses the synchronization mode that you set on the primary. The only exception to this is if you used the deprecated method of setting up HADR without using the **hadr_target_list** parameter. If you use this method, the settings for **hadr_syncmode** need to be identical on the primary and the standby.

hadr_target_list - HADR target list database configuration parameter

This parameter specifies a list of target *host:port* pairs that represent HADR standby databases. You must set the **hadr_target_list** parameter to enable multiple standby databases or to set up HADR in a Db2 pureScale environment.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

Configurable online if the environment does not use the Db2 pureScale Feature, subject to the restrictions later in this topic

Default

NULL

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_target_list** is set to the value on member 0.

The *host:port* pairs that you specify for the **hadr_target_list** configuration parameter for a primary determine which hosts act as standbys for that primary. The *host:port* pairs in the target list of a standby identify the standby hosts to be used if this standby takes over as the new HADR primary database.

As with the **hadr_remote_host** and **hadr_local_host** database configuration parameters, you can specify a host for the **hadr_target_list** configuration parameter with either a host name or an IP address. A host name can contain alphanumeric characters, dashes, and underscores only. As with the **hadr_remote_svc** and **hadr_local_svc** configuration parameters, you can specify a port for the **hadr_target_list** configuration parameter with either a port number or a service name. A service name can consist of any characters. A host name is mapped to an IP address, and a service name is mapped to a port number. The values that you specify for the **hadr_target_list** configuration parameter can be a combination of host names, host IP addresses, service names, and port values.

Separate the *host:port* pairs with the vertical bar (|), as shown in the following example:

```
host1:port1|host2:port2|host3:port3
```

To differentiate the IP part from the port part, you must enclose Numerical Internet Protocol version 6 (IPv6) addresses in square brackets: []. An example follows:

```
[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]:4000
```

In a Db2 pureScale environment, you specify a group of member hosts for the standby cluster in braces: {}. The braces are required in Db2 pureScale environment, even if only one address is listed for a cluster. In other environments, it can be omitted if there is only one address for a database. An example follows:
{host1:1000|host2:1000|host3:1000}

In the target list for a standby cluster, only a subset of the cluster's members are required. In an extreme case, only one member's address is listed. However, this is not a best practice because it creates a single point of failure. If the listed member is offline, the cluster cannot be reached. For a small cluster (such as one with four members), it is recommended that you list all members. For large clusters, list the members that are most likely to be online. It is strongly recommended that you list the preferred replay member.

Note: Currently for Db2 pureScale setups, you can specify only one cluster in the target list.

The first database in the target list of an HADR database is designated as the *principal HADR standby database*, and any other entries are called *auxiliary HADR standby databases*. You can configure the primary-principal standby HADR pair to use any synchronization mode supported in the Db2 environment by using the **hadr_syncmode** configuration parameter on the primary. The synchronization mode of the auxiliary standby targets is always SUPERASYNC. When you start HADR on the primary, the values for the **hadr_remote_host** and **hadr_remote_svc** configuration parameters are automatically set to those of the principal standby unless you set the **DB2_HADR_NO_IP_CHECK** registry variable to ON.

There is no requirement for reciprocal principal standby settings. For example, database A can designate database B as its principal standby, and database B can designate database C as its principal standby. However, general primary-standby reciprocal setting is required. For example, if database B is listed in the target list of database A, database A must be listed in the target list of database B. This ensures that after a role switch, the old primary is still accepted as the new standby of the new primary. For example, the following configuration is acceptable; however, database C is only valid standby if database B is the primary:

- On database A: **hadr_target_list** contains database B
- On database B: **hadr_target_list** contains database C and A
- On database C: **hadr_target_list** contains database B

Although you can update the value of the **hadr_target_list** parameter and have the changes take effect while the database is online, there are some restrictions if HADR is active:

- You cannot change the principal standby of the primary without first stopping HADR on the primary.
- You cannot remove a standby from the list if it is connected to the primary. To disconnect a standby, deactivate it. Then, you can remove it from the primary's target list.
- You cannot dynamically update the **hadr_target_list** configuration parameter for a standby unless you enabled the HADR reads on standby feature.
- You cannot remove the primary database from the target list of a standby if the standby is connected to the primary.
- The target list must contain IP addresses that are either IPv4 or IPv6, but not a combination of the two.

- You cannot dynamically update the **hadr_target_list** configuration parameter if you are using the Db2 pureScale Feature.

hadr_timeout - HADR timeout value

This parameter specifies the time (in seconds) that the high availability disaster recovery (HADR) process waits before considering a communication attempt to have failed.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients

Parameter type

- Configurable¹⁴

Default [range]

120 [1 - 4 294 967 295]

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **hadr_timeout** is set to the value on member 0.

indexrec - Index re-creation time

This parameter indicates when the database manager attempts to rebuild invalid indexes, and whether or not any index build is redone during rollforward or high availability disaster recovery (HADR) log replay on the standby database.

Configuration type

Database and Database Manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

UNIX Database Manager

restart [restart; restart_no_redo; access; access_no_redo]

Windows Database Manager

restart [restart; restart_no_redo; access; access_no_redo]

Database

Use system setting [system; restart; restart_no_redo; access; access_no_redo]

14. Changes to this parameter take effect on database activation. If the database is already online, you can have changes take effect by stopping and restarting HADR on the primary database.

There are five possible settings for this parameter:

SYSTEM

Use system setting specified in the database manager configuration file to decide when invalid indexes are rebuilt, and whether any index build log records are to be redone during rollforward or HADR log replay.

Note: This setting is only valid for database configurations.

ACCESS

Invalid indexes are rebuilt when the underlying table is first accessed. Any fully logged index builds are redone during rollforward or HADR log replay. When HADR is started and an HADR takeover occurs, any invalid indexes are rebuilt after takeover when the underlying table is first accessed.

ACCESS_NO_REDO

Invalid indexes are rebuilt when the underlying table is first accessed. Any fully logged index build is not redone during rollforward or HADR log replay and those indexes are left invalid. When HADR is started and an HADR takeover takes place, any invalid indexes are rebuilt after takeover when the underlying table is first accessed. Access to the underlying tables on the new primary cause index rebuild, which causes log records to be written and then sent to the new standby, which in turn causes the indexes to be invalidated on the standby.

RESTART

The default value for **indexrec**. Invalid indexes are rebuilt when a **RESTART DATABASE** command is either explicitly or implicitly issued. Any fully logged index build will be redone during rollforward or HADR log replay. When HADR is started and an HADR takeover takes place, any invalid indexes will be rebuilt at the end of takeover.

Note: In a Db2 pureScale environment, indexes are rebuilt only during a group crash recovery, not as part of member crash recovery.

Note: When a database terminates abnormally while applications are connected to it, and the **autorestart** parameter is enabled, a **RESTART DATABASE** command is implicitly issued when an application connects to a database. If the database terminates normally, then the **RESTART DATABASE** command must be issued before any connections are made to the database, otherwise index recreation does not take place. If the command is not issued, the invalid indexes are rebuilt the next time the underlying table is accessed.

RESTART_NO_REDO

Invalid indexes are rebuilt when a **RESTART DATABASE** command is either explicitly or implicitly issued. Any fully logged index build is not redone during rollforward or HADR log replay and instead those indexes are rebuilt when rollforward completes or when HADR takeover takes place. Takeover causes index rebuild on underlying tables on the new primary, which causes log records to be written and then sent to the new standby, which in turn causes the indexes to be invalidated on the standby.

When a database terminates abnormally while applications are connected to it, and the **autorestart** parameter is enabled, a **RESTART DATABASE** command is implicitly issued when an application connects to a database. If the database terminates normally, then the **RESTART DATABASE** command must be issued before any connections are made to the database, otherwise

index recreation does not take place. If the command is not issued, the invalid indexes are rebuilt the next time the underlying table is accessed.

Indexes can become invalid when fatal disk problems occur. If this happens to the data itself, the data could be lost. However, if this happens to an index, the index can be recovered by recreating it. If an index is rebuilt while users are connected to the database, two problems could occur:

- An unexpected degradation in response time might occur as the index file is re-created. Users accessing the table and using this particular index would wait while the index was being rebuilt.
- Unexpected locks might be held after index recreation, especially if the user transaction that caused the index to be re-created never performed a COMMIT or ROLLBACK.

Recommendation: The best choice for this option on a high-user server and if restart time is not a concern, would be to have the index rebuilt at **DATABASE RESTART** time as part of the process of bringing the database back online after a crash.

Setting this parameter to ACCESS or to ACCESS_NO_REDO result in a degradation of the performance of the database manager while the index is being re-created. Any user accessing that specific index or table would have to wait until the index is re-created.

If this parameter is set to RESTART, the time taken to restart the database is longer due to index re-creation, but normal processing would not be impacted once the database has been brought back online.

The difference between the RESTART and the RESTART_NO_REDO values, or between the ACCESS and the ACCESS_NO_REDO values, is only significant when full logging is activated for index build operations, such as **CREATE INDEX** and **REORG INDEX** operations, or for an index rebuild. You can activate logging by enabling the **Logindexbuild** database configuration parameter or by enabling LOG INDEX BUILD when altering a table. By setting **indexrec** to either RESTART or ACCESS, operations involving a logged index build can be rolled forward without leaving the index object in an invalid state, which would require the index to be rebuilt at a later time.

locklist - Maximum storage for lock list

This parameter indicates the amount of storage that is allocated to the lock list. There is one lock list per database and it contains the locks held by all applications concurrently connected to the database.

Configuration type

Database

Parameter type

- Configurable
- Configurable by member in a Db2 pureScale environment

Default [range]

Automatic [4 - 134217728]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

When the first application connects to the database

When freed

When last application disconnects from the database

Locking is the mechanism that the database manager uses to control concurrent access to data in the database by multiple applications. Both rows and tables can be locked. The database manager can also acquire locks for internal use.

When this parameter is set to `AUTOMATIC`, it is enabled for self tuning. This allows the memory tuner to dynamically size the memory area controlled by this parameter as the workload requirements change. Because the memory tuner trades memory resources between different memory consumers, there must be at least two memory consumers enabled for self tuning in order for self tuning to be active

Although the value of **locklist** can be tuned together with the **maxlocks** parameter, disabling self tuning of the **locklist** parameter does not automatically disable self tuning of the **maxlocks** parameter. Enabling self tuning of the **locklist** parameter automatically enables self tuning of the **maxlocks** parameter.

Automatic tuning of this configuration parameter will only occur when self tuning memory is enabled for the database (the **self_tuning_mem** configuration parameter is set to `ON`.)

On all platforms, each lock requires 128 or 256 bytes of the lock list, depending on whether other locks are held on the object:

- 256 bytes are required to hold a lock on an object that has no other locks held on it
- 128 bytes are required to record a lock on an object that has an existing lock held on it.

When the percentage of the lock list used by one application reaches **maxlocks**, the database manager will perform lock escalation, from row to table, for the locks held by the application. This calculation is an approximation, assuming shared locks only. The percentage of the lock list used is calculated by multiplying the number of locks held by the application by the value required to hold a lock on an object that has other locks held on it. Although the escalation process itself does not take much time, locking entire tables (versus individual rows) decreases concurrency, and overall database performance might decrease for subsequent accesses against the affected tables. Suggestions of how to control the size of the lock list are:

- Perform frequent `COMMITs` to release locks.
- When performing many updates, lock the entire table before updating (using the `SQL LOCK TABLE` statement). This will use only one lock, keeps others from interfering with the updates, but does reduce concurrency of the data.

You can also use the `LOCKSIZE` option of the `ALTER TABLE` statement to control how locking is done for a specific table.

Use of the Repeatable Read isolation level might result in an automatic table lock.

- Use the Cursor Stability isolation level when possible to decrease the number of share locks held. If application integrity requirements are not compromised use Uncommitted Read instead of Cursor Stability to further decrease the amount of locking.
- Set **locklist** to AUTOMATIC. The lock list will increase synchronously to avoid lock escalation or a lock list full situation.

Once the lock list is full, performance can degrade since lock escalation will generate more table locks and fewer row locks, thus reducing concurrency on shared objects in the database. Additionally there might be more deadlocks between applications (since they are all waiting on a limited number of table locks), which will result in transactions being rolled back. Your application will receive an SQLCODE of -912 when the maximum number of lock requests has been reached for the database.

Recommendation: If lock escalations are causing performance concerns you might need to increase the value of this parameter or the **maxlocks** parameter. You can use the database system monitor to determine if lock escalations are occurring. Refer to the **lock_escals** (lock escalations) monitor element.

The following steps might help in determining the number of pages required for your lock list:

1. Calculate a lower bound for the size of your lock list, using *one* of the following calculations, depending on your environment:

a.

$$(512 * 128 * \text{maxapps}) / 4096$$

b. with Concentrator enabled:

$$(512 * 128 * \text{max_coordagents}) / 4096$$

c. in a partitioned database with Concentrator enabled:

$$(512 * 128 * \text{max_coordagents} * \text{number of database partitions}) / 4096$$

where 512 is an estimate of the average number of locks per application and 128 is the number of bytes required for each lock against an object that has an existing lock.

2. Calculate an upper bound for the size of your lock list:

$$(512 * 256 * \text{maxapps}) / 4096$$

where 256 is the number of bytes required for the first lock against an object.

Note: While in a Db2 pureScale environment, set the **locklist** configuration parameter to be equal to the upper bound of the value calculated in this step and 3% of the total number of pages for all of the buffer pools existing in the currently connected database.

3. Estimate the amount of concurrency you will have against your data and based on your expectations, choose an initial value for **locklist** that falls between the upper and lower bounds that you have calculated.
4. Using the database system monitor, as described in the following paragraph, tune the value of this parameter.

If **maxapps** or **max_coordagents** are set to AUTOMATIC in your applicable scenario, you should also set **locklist** to AUTOMATIC.

You can use the database system monitor to determine the maximum number of locks held by a given transaction. Refer to the **locks_held_top** (maximum number of locks held) monitor element.

This information can help you validate or adjust the estimated number of locks per application. In order to perform this validation, you will have to sample several applications, noting that the monitor information is provided at a transaction level, not an application level.

You might also want to increase **locklist** if **maxapps** is increased, or if the applications being run perform infrequent commits.

You should consider rebinding applications (using the **REBIND** command) after changing this parameter.

locktimeout - Lock timeout

This parameter specifies the number of seconds that an application will wait to obtain a lock, helping avoid global deadlocks for applications.

Configuration type

Database

Parameter type

- Configurable

Default [range]

-1 [-1; 0 - 32 767]

Unit of measure

Seconds

If you set this parameter to 0, locks are not waited for. In this situation, if no lock is available at the time of the request, the application immediately receives a -911.

If you set this parameter to -1, lock timeout detection is turned off. In this situation a lock will be waited for (if one is not available at the time of the request) until either of the following events occur:

- The lock is granted
- A deadlock occurs.

Note: The value you specify for this configuration parameter is not used to control lock timeouts for event monitor target tables. Event monitors use a separate, non-configurable setting that will cause locks on event monitor tables to time out.

Recommendation: In a transaction processing (OLTP) environment, you can use an initial starting value of 30 seconds. In a query-only environment you could start with a higher value. In both cases, you should use benchmarking techniques to tune this parameter.

The value should be set to quickly detect waits that are occurring because of an abnormal situation, such as a transaction that is stalled (possibly as a result of a user leaving their workstation). You should set it high enough so valid lock requests do not time out because of peak workloads, during which time, there is more waiting for locks.

You can use the database system monitor to help you track the number of times an application (connection) experienced a lock timeout or that a database detected a timeout situation for all applications that were connected.

High values of the **lock_timeout** (number of lock timeouts) monitor element can be caused by:

- Too low a value for this configuration parameter.
- An application (transaction) that is holding locks for an extended period. You can use the database system monitor to further investigate these applications.
- A concurrency problem, that could be caused by lock escalations (from row-level to a table-level lock).

log_appl_info - Application information log record database configuration parameter

This parameter specifies that the application information log record is written at the start of each update transaction.

Configuration type

Database

Parameter type

Configurable

Default [range]

No [Yes; No]

When set to YES, an extra log record is added at the start of each update transaction. The application information log record is used for the transaction (refer to "Application information log record" in Transaction manager log records).

Enable the **log_appl_info** configuration parameter if there are tables created with the DATA CAPTURE CHANGES option for replication purpose (or other purpose such as auditing).

log_ddl_stmts - Log Data Definition Language (DDL) statements database configuration parameter

This parameter specifies that extra information regarding Data Definition Language (DDL) statements will be written to the log.

Configuration type

Database

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

No [Yes; No]

The default setting of the **log_ddl_stmts** configuration parameter avoids the overhead of writing extra log records for DDL operations. Set this parameter to YES if you need to replicate DDL operations and a replication capture program is used to capture changes from the log.

log_retain_status - Log retain status indicator

If the **logarchmeth1** database configuration parameter is set to **logretain** then the log retain status parameter will show a value of RECOVERY, otherwise it will show a value of NO.

Configuration type

Database

Parameter type
Informational

logarchcompr1 - Primary archived log file compression configuration parameter

This parameter specifies whether the log files written to the primary archive destination for logs are compressed.

Configuration type
Database

Applies to

- Database servers with local and remote clients
- Clients
- Database servers with local clients
- Partitioned database servers with local and remote clients

Parameter type
Configurable online

Default [range]
OFF [ON; NX842]

- OFF** Specifies that log files written to the primary archive destination for logs are not compressed.
- ON** Specifies that log files written to the primary archive destination for logs are compressed. If set dynamically, log files already archived are not compressed.
- NX842** Specifies that log files written to the primary archive destination for logs are compressed using the NX842 hardware compression method. If set dynamically, log files already archived are not compressed. This feature is only available on AIX.

Note:

- If you set the **logarchmeth1** configuration parameter to a value other than DISK, TSM, or VENDOR, log archive compression has no effect regardless of the **logarchcompr1** configuration parameter setting.

logarchcompr2 - Secondary archived log file compression configuration parameter

This parameter specifies whether the log files written to the secondary archive destination for logs are compressed.

Configuration type
Database

Applies to

- Database servers with local and remote clients
- Clients
- Database servers with local clients
- Partitioned database servers with local and remote clients

Parameter type
Configurable online

Default [range]

OFF [ON; NX842]

OFF Specifies that log files written to the secondary archive destination for logs are not compressed.

ON Specifies that log files written to the secondary archive destination for logs are compressed. If set dynamically, log files already archived are not compressed.

NX842 Specifies that log files written to the primary archive destination for logs are compressed using the NX842 hardware compression method. If set dynamically, log files already archived are not compressed. This feature is only available on AIX.

Note:

- If you set the **logarchmeth2** configuration parameter to a value other than DISK, TSM, or VENDOR, log archive compression has no effect regardless of the **logarchcompr2** configuration parameter setting.

logarchmeth1 - Primary log archive method

You can use this parameter to specify the media type of the primary destination for logs that are archived from the current log path.

Configuration type

Database

Applies to

- Database servers with local and remote clients
- Clients
- Database servers with local clients
- Partitioned database servers with local and remote clients

Parameter type

Configurable Online.

Default [range]

OFF [LOGRETAIN, USEREXIT, DISK, TSM, VENDOR]

The value that you set for the **logarchmeth1** configuration parameter depends on your database environment and your database login requirements.

OFF Specifies that the log archiving method is not used. If you set both the **logarchmeth1** and **logarchmeth2** configuration parameters to OFF, which is the default, the database is considered to be using circular logging and is not rollforward recoverable.

LOGRETAIN

Specifies that active log files are retained and become online archive log files for use in rollforward recovery.

USEREXIT

Specifies that log retention logging is performed and that a user exit program must be used to archive and retrieve the log files. Log files are archived when they are full. They are retrieved when the **ROLLFORWARD** utility must use them to restore a database.

DISK

You must follow this value with a colon (:) and then a fully qualified existing path name where the log files are archived. For example, if you set

the **logarchmeth1** configuration parameter to DISK:/u/dbuser/archived_logs, the archive log files are placed in the /u/dbuser/archived_logs/instance/dbname/nodename/logstream/chainid/ directory.

On Windows operating systems, you must format the path name by using UNC disk addressing, such as \\u\dbuser\archived_logs\, or drive letters, such as C:\u\dbuser\archived_logs.

Note: If you are archiving to tape, you can use the **db2tapemgr** utility to store and retrieve log files.

TSM If specified without any additional configuration parameters, indicates that log files are archived on the local TSM server by using the default management class. If you follow this option with a colon (:) and a TSM management class, the log files are archived by using the specified management class.

When you are archiving logs by using TSM, before you use the management class that is specified by the database configuration parameter, TSM attempts to bind the object to the management class that you specified in the INCLUDE-EXCLUDE list in the TSM client options file. If a match is not found, the default TSM management class that you specified on the TSM server is used. TSM then rebinds the object to the management class that you specified for the database configuration parameter. Thus, the default management class and the management class that you specify for the database configuration parameter must contain an archive copy group, or the archive operation fails. Examples of TSM entries are:

- With a management class specified - db2 update db cfg for mydb using logarchmeth1 TSM:DB2_LOGS
- With no management class specified - db2 update db cfg for mydb using logarchmeth1 TSM

VENDOR Specifies that a vendor library is used to archive the log files. You must follow this value with a colon (:) and the name of the library. The APIs in the library must use the backup and restore APIs for vendor products. An example of a vendor entry is - db2 update db cfg for mydb using logarchmeth1 VENDOR:/home/dbuser/vendorLib/<library name>

Note:

- If you set either the **logarchmeth1** or **logarchmeth2** configuration parameter to a value other than OFF, the database is configured for rollforward recovery.
- If you use the userexit or logretain option for the **logarchmeth1** configuration parameter, you must set the **logarchmeth2** configuration parameter to OFF.
- To assign an archive path with a space in it, use the **db2CfgSet** API.
- The archive log path must not contain log files that do not belong to the current database. If the archive log path was previously used for a database of the same name, these log files must be removed before you use the current archive log path.

logarchmeth2 - Secondary log archive method

This parameter specifies the media type of the secondary destination for logs that are archived from either the current log path or the mirror log path.

Configuration type

Database

Applies to

- Database servers with local and remote clients
- Clients
- Database servers with local clients
- Partitioned database servers with local and remote clients

Parameter type

Configurable online

Default [range]

OFF [DISK, TSM, VENDOR]

OFF Specifies that the log archiving method is not used. If you set both **logarchmeth1** and **logarchmeth2** configuration parameters to OFF, the database is considered to be using circular logging and is not rollforward recoverable. This is the default.

DISK You must follow this value with a colon (:) and then a fully qualified existing path name where the log files will be archived. For example, if you set the **logarchmeth1** configuration parameter to DISK:/u/dbuser/archived_logs the archive log files are placed in the /u/dbuser/archived_logs/*instance/dbname/nodename/chainid/* directory.

Note: If you are archiving to tape, you can use the **db2tapemgr** utility to store and retrieve log files.

TSM If specified without any additional configuration parameters, indicates that log files are archived on the local TSM server by using the default management class. If you follow this option with a colon (:) and a TSM management class, the log files are archived using the specified management class.

When archiving logs using TSM, before using the management class that is specified by the database configuration parameter, TSM attempts to bind the object to the management class that you specified in the INCLUDE-EXCLUDE list in the TSM client options file. If a match is not found, the default TSM management class that you specified on the TSM server is used. TSM then rebinds the object to the management class that you specified for the database configuration parameter. Thus, the default management class and the management class that you specify for the database configuration parameter must contain an archive copy group, or the archive operation fails.

VENDOR

Specifies that a vendor library is used to archive the log files. You must follow this value with a colon (:) and the name of the library. The APIs in the library must use the backup and restore APIs for vendor products.

Notes:

- If you set either the **logarchmeth1** or **logarchmeth2** configuration parameter to a value other than OFF, the database is configured for rollforward recovery.
- If you use the **userexit** or **logretain** option for the **logarchmeth1** configuration parameter, you must set the **logarchmeth2** configuration parameter to OFF.

- The archive log path must not contain log files that do not belong to the current database. If the archive log path was previously used for a database of the same name, these log files must be removed before using the current archive log path.

If you use the **logarchmeth2** configuration parameter to specify a path, log files will be archived to both this destination and the destination that is specified by the **logarchmeth1** database configuration parameter. Which log files the **logarchmeth2** configuration parameter archives depends on whether you also set the value of the **mirrorlogpath** database configuration parameter:

- If you do not set the value of the **mirrorlogpath** configuration parameter, the **logarchmeth2** configuration parameter archives log files from the current log path, as specified by the **logpath** configuration parameter.
- If you set the value of the **mirrorlogpath** configuration parameter, the **logarchmeth2** configuration parameter archives log files from the mirror log path.

logarchopt1 - Primary log archive options

This parameter specifies a string of options which control log archiving behavior when the primary archived log method, **logarchmeth1**, is enabled. Multiple options can be specified in the string and must be separated by white space.

See the section that is titled *Log archive options 1 (logarchopt1), log archive options 2 (logarchopt2)* of Configuration parameters for database logging for a complete description of log archive options.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Default [range]

Null [not applicable]

Restrictions

See Configuration parameters for database logging for a complete description of logarchopt1 and logarchopt2 options, including restrictions.

Note: Only the first 30 characters are stored in the history file. If AUTO_DEL_REC is configured to drive automatic log file deletion, the original value might not be passed correctly to the shared library and this can cause the deletion to fail.

logarchopt2 - Secondary log archive options

This parameter specifies a string of options which control log archiving behavior when the secondary archived log method, **logarchmeth2**, is enabled. Multiple options can be specified in the string and must be separated by white space.

See the section that is titled *Log archive options 1 (logarchopt1), log archive options 2 (logarchopt2)* of Configuration parameters for database logging for a complete description of log archive options.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Default [range]

Null [not applicable]

Restrictions

See Configuration parameters for database logging for a complete description of logarchopt1 and logarchopt2 options, including restrictions.

Note: Only the first 30 characters are stored in the history file. If AUTO_DEL_REC is configured to drive automatic log file deletion, the original value might not be passed correctly to the shared library and this can cause the deletion to fail.

logbufsz - Log buffer size

This parameter allows you to specify the amount of the database heap (defined by the **dbheap** parameter) to use as a buffer for log records before writing these records to disk.

Configuration type

Database

Parameter type

- Configurable
- Configurable by member in a Db2 pureScale environment

Default [range]

32-bit platforms

256 [4 - 4 096]

64-bit platforms

256 [4 - 131 070]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

Log records are written to disk when one of the following occurs:

- A transaction commits or a group of transactions commit, as defined by the **mincommit** configuration parameter
- The log buffer is full
- As a result of some other internal database manager event.

This parameter must also be less than or equal to the **dbheap** parameter. Buffering the log records will result in more efficient logging file I/O because the log records will be written to disk less frequently and more log records will be written at each time.

Recommendation: Increase the size of this buffer area if there is considerable read activity on a dedicated log disk, or there is high disk utilization. When increasing the value of this parameter, you should also consider the **dbheap** parameter since the log buffer area uses space controlled by the **dbheap** parameter.

You can use the database system monitor to determine how often the log buffer has become full, requiring it to be written to disk before new log records can be written. Refer to the **num_log_buffer_full** (number of full log buffers) monitor element.

logfilsiz - Size of log files

This parameter defines the size of each primary and secondary log file. The size of these log files limits the number of log records that can be written to them before they become full and a new log file is required.

Configuration type

Database

Parameter type

Configurable

Default [range]

1000 [4 - 16 777 152]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

The use of primary and secondary log files as well as the action taken when a log file becomes full are dependent on the type of logging that is being performed:

- Circular logging

A primary log file can be reused when the changes recorded in it have been committed. If the log file size is small and applications have processed a large number of changes to the database without committing the changes, a primary log file can quickly become full. If all primary log files become full, the database manager will allocate secondary log files to hold the new log records.

- Log retain or archive logging

When a log file is full, it is closed and not overwritten. If archive logging is configured then the log file will be subsequently archived.

Recommendation: You must balance the size of the log files with the number of log files:

- The value of the **logfilsiz** should be increased if the database has a large number of update, delete, or insert transactions running against it which will cause the log file to become full very quickly.

Note: The upper limit of log file size, combined with the upper limit of the number of log files (**logprimary** + **logsecond**), gives an upper limit of 1024 GB of active log space.

A log file that is too small can affect system performance because of the processing time of archiving old log files, allocating new log files, and waiting for a usable log file.

- The value of the **logfilesiz** should be reduced if disk space is scarce, since primary logs are preallocated at this size.

A log file that is too large can reduce your flexibility when managing archived log files and copies of log files, since some media might not be able to hold an entire log file.

Also, the size of the log file can impact the frequency of which log files are archived and the amount of time it takes to archive an individual log file. These factors will impact the availability of log files in the archive location for disaster recovery scenarios if something happens to the primary server. The **ARCHIVE LOG FOR DATABASE** command can be used to truncate and archive log files on a more frequent basis if necessary.

If you are using log retention, the current active log file is closed and truncated when the last application disconnects from a database. When the next connection to the database occurs, the next log file is used. Therefore, if you understand the logging requirements of your concurrent applications, you might be able to determine a log file size that will not allocate excessive amounts of wasted space.

loghead - First active log file

This parameter contains the name of the log file that is currently active.

Configuration type

Database

Parameter type

Informational

logindexbuild - Log index pages created

This parameter specifies whether index creation, recreation, or reorganization operations are logged so that indexes can be reconstructed during Db2 rollforward operations or high availability disaster recovery (HADR) log replay procedures.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]

Off [On, Off]

When you set the **logindexbuild** configuration parameter to Off, index objects are marked invalid while rolling forward through index creation, recreation, or reorganization operations. You must re-create indexes after the rolling forward

operation completes and the database is restarted. Index re-creation depends on the value of the **indexrec** configuration parameter and if you issue an explicit **RESTART DATABASE** command.

When you set the **logindexbuild** configuration parameter to 0n, each index page is logged during index creation, recreation and reorganization. Index objects are not marked invalid, and you do not have to re-create indexes after the rolling forward operation completes.

logpath - Location of log files

This parameter contains the current path being used for logging purposes.

Configuration type

Database

Parameter type

Informational

You cannot change this parameter directly, because it is set by the database manager after a change to the **newlogpath** parameter takes effect.

The default log path for a member is a directory within the global database directory. For example, given an instance name of dbinst, a database name of dbname, and a user named dbuser, the default log path for a member might be /home/dbuser/dbinst/NODE0000/SQL00001/LOGSTREAM0000. In a Db2 pureScale environment, there is one LOGSTREAMxxxx directory for each member.

Important: If the current log path is not usable, Db2 switches back to the default path. This feature allows the database to come up.

logprimary - Number of primary log files

This parameter allows you to specify the number of primary log files to be preallocated. The primary log files establish a fixed amount of storage allocated to the recovery log files.

Configuration type

Database

Parameter type

Configurable

Default [range]

3 [2 - 256]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Counter

When allocated

- The database is created
- A log is moved to a different location (which occurs when the **newlogpath** parameter is updated)
- When the first log record is written after the database is next started following an increase in the value of this parameter (**logprimary**), provided that the database is not started as an HADR standby database

- A log file becomes inactive and a new log file is required to keep at least **logprimary** logs in the primary log path (the **logarchmeth1** or **logarchmeth2** parameter must not be set to a value other than OFF).
- If the **logfilesiz** parameter has been changed, the log files are re-sized during the next database startup, provided that it is not started as an HADR standby database

When freed

Not freed unless this parameter decreases. If decreased, unneeded log files are deleted during the next connection to the database.

Under circular logging, the primary logs are used repeatedly in sequence. That is, when a log is full, the next primary log in the sequence is used if it is available. A log is considered available if all units of work with log records in it have been committed or rolled-back. If the next primary log in sequence is not available, then a secondary log is allocated and used. Additional secondary logs are allocated and used until the next primary log in the sequence becomes available or the limit imposed by the **logsecond** parameter is reached. These secondary log files are dynamically deallocated as they are no longer needed by the database manager.

The number of primary and secondary log files must comply with the following:

- If **logsecond** has a value of -1, **logprimary** \leq 256.
- If **logsecond** does not have a value of -1, **(logprimary + logsecond)** \leq 256.

Recommendation: The value chosen for this parameter depends on a number of factors, including the type of logging being used, the size of the log files, and the type of processing environment (for example, length of transactions and frequency of commits).

Increasing this value will increase the disk requirements for the logs because the primary log files are preallocated during the very first connection to the database.

If you find that secondary log files are frequently being allocated, you might be able to improve system performance by increasing the log file size (**logfilesiz**) or by increasing the number of primary log files.

For databases that are not frequently accessed, in order to save disk storage, set the parameter to 2. For databases enabled for rollforward recovery, set the parameter larger to avoid the additional processing time of allocating new logs almost immediately.

You can use the database system monitor to help you size the primary log files. Observation of the following monitor values over a period of time will aid in better tuning decisions, as average values might be more representative of your ongoing requirements.

- **sec_log_used_top** (maximum secondary log space used)
- **tot_log_used_top** (maximum total log space used)
- **sec_logs_allocated** (secondary logs allocated currently)

logsecond - Number of secondary log files

This parameter specifies the number of secondary log files that are created and used for recovery log files. The secondary log files are created only as needed.

Configuration type

Database

Parameter type
Configurable Online

Propagation class
Immediate

Default [range]
10 [-1; 0 - 254]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure
Counter

When allocated
As needed when **logprimary** is insufficient. For more allocation details, see the details that follow.

When freed
Over time as the database manager determines which secondary log files are no longer needed.

When the primary log files become full, the secondary log files (of size **logfilsiz**) are allocated one at a time as needed, up to a maximum number as controlled by this parameter. If more secondary log files are required than are allowed by this parameter, an error code will be returned to the application.

If you set **logsecond** to -1, the database is configured with infinite active log space. There is no limit on the size or the number of in-flight transactions running on the database. If you set **logsecond** to -1, you still use the **logprimary** and **logfilsiz** configuration parameters to specify how many log files the database manager should keep in the active log path. If the database manager needs to read log data from a log file, but the file is not in the active log path, the database manager retrieves the log file from the archive to the active log path. (The database manager retrieves the files to the overflow log path, if you have configured one.) Once the log file is retrieved, the database manager will cache this file in the active log path so that other reads of log data from the same file will be fast. The database manager will manage the retrieval, caching, and removal of these log files as required.

Note: You cannot configure infinite active log space in Db2 pureScale environments.

If your log path is a raw device, you must configure the **overflowlogpath** configuration parameter in order to set **logsecond** to -1.

By setting **logsecond** to -1, you will have no limit on the size of the unit of work or the number of concurrent units of work. However, rollback (both at the savepoint level and at the unit of work level) could be very slow due to the need to retrieve log files from the archive. Crash recovery could also be very slow for the same reason. The database manager writes a message to the administration notification log to warn you that the current set of active units of work has exceeded the primary log files. This is an indication that rollback or crash recovery could be extremely slow.

To set **logsecond** to -1, the **logarchmeth1** configuration parameter must be set to a value other than OFF or LOGRETAIN.

Recommendation: Use secondary log files for databases that have periodic needs for large amounts of log space. For example, an application that is run once a month might require log space beyond that provided by the primary log files. Because secondary log files do not require permanent file space, they are advantageous in this type of situation.

When infinite logging is enabled (**logsecond** to -1), the database manager does not reserve active log space for transactions that might need to roll back and write log records. During rollback processing, if both the active log path and archive target are full (or if the archive target is inaccessible), then the **blk_log_dsk_ful** (block on log disk full db configuration parameter) should also be ENABLED to avoid database failures.

max_log - Maximum log per transaction

This parameter specifies if there is a limit to the percentage of the primary log space that a transaction can consume, and what that limit is.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

0 [0 - 100]

Unit of measure

Percentage

If the value is not 0, this parameter indicates the percentage of primary log space that can be consumed by one transaction.

If the value is set to 0, there is no limit to the percentage of total primary log space that a transaction can consume.

If an application violates the **max_log** configuration, the application is forced to disconnect from the database and the transaction is rolled back.

You can override this behavior by setting the **DB2_FORCE_APP_ON_MAX_LOG** registry variable to "FALSE". This will cause transactions that violate the **max_log** configuration to fail; however, the application can still commit the work completed by previous statements in the unit or work, or it can roll the completed work back to undo the unit of work.

This parameter, along with the **num_log_span** configuration parameter, can be useful when enabling infinite active logspace. If infinite logging is on (that is, if **logsecond** is set to -1) then transactions are not restricted to the upper limit of the number of log files (**logprimary** + **logsecond**). When the value of **logprimary** is reached, Db2 starts to archive the active logs, rather than failing the transaction. This can cause problems if, for example, an application contains a long running transaction that is left uncommitted. If this occurs, the active logspace continues to

grow, possibly leading to poor crash recovery performance. To prevent this, you can specify values for one or both of the **max_log** or **num_log_span** configuration parameters.

Note: The following Db2 commands are excluded from the limitation imposed by the **max_log** configuration parameter: ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG, RESTORE DATABASE, and ROLLFORWARD DATABASE.

maxapps - Maximum number of active applications

This parameter specifies the maximum number of concurrent applications that can be connected (both local and remote) to a database. Since each application that attaches to a database causes some private memory to be allocated, allowing a larger number of concurrent applications will potentially use more memory.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

Automatic [1 - 60 000]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Counter

Setting **maxapps** to `automatic` has the effect of allowing any number of connected applications. The database manager will dynamically allocate the resources it needs to support new applications.

If you do not want to set this parameter to `automatic`, the value of this parameter must be equal to or greater than the sum of the connected applications, plus the number of these same applications that might be concurrently in the process of completing a two-phase commit or rollback. Then add to this sum the anticipated number of indoubt transactions that might exist at any one time.

When an application attempts to connect to a database, but **maxapps** has already been reached, an error is returned to the application indicating that the maximum number of applications have been connected to the database.

In a partitioned database environment, this is the maximum number of applications that can be concurrently active against a database partition. This parameter limits the number of active applications against the database partition on a database partition server, regardless of whether the server is the coordinator node for the application or not. The catalog node in a partitioned database environment requires a higher value for **maxapps** than is the case for other types of environments because, in the partitioned database environment, every application requires a connection to the catalog node.

Recommendation: Increasing the value of this parameter without lowering the **maxlocks** parameter or increasing the **locklist** parameter could cause you to reach the database limit on locks (**locklist**) rather than the application limit and as a result cause pervasive lock escalation problems.

To a certain extent, the maximum number of applications is also governed by **max_coordagents**. An application can only connect to the database, if there is an available connection (**maxapps**) as well as an available coordinating agent (**max_coordagents**).

maxfilop - Maximum database files open per database

This parameter specifies the maximum number of file handles that can be open per database. Each active application is counted towards the value specified by **maxfilop**.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Transaction boundary

Default [range]

AIX, Sun, HP, and Linux 64-bit

61 440 [64 - 61 440]

Linux 32-bit

30 720 [64 - 30 720]

Windows 32-bit

32 768 [64 - 32 768]

Windows 64-bit

65 335 [64 - 65 335]

Unit of measure

Counter

If opening a file causes this value to be exceeded, some files in use by this database are closed. If **maxfilop** is too small, the additional processing time of opening and closing files will become excessive and might degrade performance.

Both SMS table spaces and DMS table space file containers are treated as files in the database manager's interaction with the operating system, and file handles are required. More files are generally used by SMS table spaces compared to the number of containers used for a DMS file table space. Therefore, if you are using SMS table spaces, you will need a larger value for this parameter compared to what you would require for DMS file table spaces.

You can also use this parameter to ensure that the overall total of file handles used by the database manager does not exceed the operating system limit by limiting the number of handles per database to a specific number; the actual number will vary depending on the number of databases running concurrently.

maxlocks - Maximum percent of lock list before escalation

This parameter defines a percentage of the lock list held by an application that must be filled before the database manager performs lock escalation.

Configuration type

Database

Parameter type

- Configurable
- Configurable by member in a Db2 pureScale environment

Default [range]

Automatic [1 - 100]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Percentage

Lock escalation is the process of replacing row locks with table locks, reducing the number of locks in the list. When the number of locks held by any one application reaches this percentage of the total lock list size, lock escalation will occur for the locks held by that application. Lock escalation also occurs if the lock list runs out of space.

The database manager determines which locks to escalate by looking through the lock list for the application and finding the table with the most row locks. If after replacing these with a single table lock, the **maxlocks** value is no longer exceeded, lock escalation will stop. If not, it will continue until the percentage of the lock list held is lower than the value of **maxlocks**. The **maxlocks** parameter multiplied by the **maxappls** parameter cannot be less than 100.

When this parameter is set to AUTOMATIC, it is enabled for self tuning. This allows the memory tuner to dynamically size the memory area controlled by this parameter as the workload requirements change. Because the memory tuner trades memory resources between different memory consumers, there must be at least two memory consumers enabled for self tuning in order for self tuning to be active.

Although the value of **locklist** can be tuned together with the **maxlocks** parameter, disabling self tuning of the **locklist** parameter does not automatically disable self tuning of the **maxlocks** parameter. Enabling self tuning of the **locklist** parameter automatically enables self tuning of the **maxlocks** parameter.

Automatic tuning of this configuration parameter will only occur when self tuning memory is enabled for the database (the **self_tuning_mem** configuration parameter is set to ON).

On all platforms, each lock requires 128 or 256 bytes of the lock list, depending on whether other locks are held on the object:

- 256 bytes are required to hold a lock on an object that has no other locks held on it.
- 128 bytes are required to record a lock on an object that has an existing lock held on it.

Recommendation: The following formula allows you to set **maxlocks** to allow an application to hold twice the average number of locks:

$$\text{maxlocks} = 2 * 100 / \text{maxapps}$$

Where 2 is used to achieve twice the average and 100 represents the largest percentage value allowed. If you have only a few applications that run concurrently, you could use the following formula as an alternative to the first formula:

$$\text{maxlocks} = 2 * 100 / (\text{average number of applications running concurrently})$$

One of the considerations when setting **maxlocks** is to use it in conjunction with the size of the lock list (**locklist**). The actual limit of the number of locks held by an application before lock escalation occurs is:

- $\text{maxlocks} * \text{locklist} * 4096 / (100 * 128)$

Where 4096 is the number of bytes in a page, 100 is the largest percentage value allowed for **maxlocks**, and 128 is the number of bytes per lock. If you know that one of your applications requires 1000 locks, and you do not want lock escalation to occur, then you should choose values for **maxlocks** and **locklist** in this formula so that the result is greater than 1000. (Using 10 for **maxlocks** and 100 for **locklist**, this formula results in greater than the 1000 locks needed.)

If **maxlocks** is set too low, lock escalation happens when there is still enough lock space for other concurrent applications. If **maxlocks** is set too high, a few applications can consume most of the lock space, and other applications will have to perform lock escalation. The need for lock escalation in this case results in poor concurrency.

You can use the database system monitor to help you track and tune this configuration parameter.

min_dec_div_3 - Decimal division scale to 3

This parameter is deprecated starting with Db2 Version 11.1.3.3 and replaced by the **dec_arithmetic** configuration parameter.

Starting with Version 11.1.3.3, the **min_dec_div_3** configuration parameter is ignored if the **dec_arithmetic** configuration parameter is set to a non-default value. Setting **dec_arithmetic** to "DEC.3" results in the same behavior as enabling the **min_dec_div_3** configuration parameter.

Note: The following information applies only to data servers and clients Version 11.1.2.2 and earlier.

This parameter is provided as a quick way to enable a change to computation of the scale for decimal division in SQL.

Configuration type

Database

Parameter type

Configurable

Default [range]

No [Yes, No]

The `min_dec_div_3` database configuration parameter changes the resulting scale of a decimal arithmetic operation involving division. It can be set to "Yes" or "No". The default value for `min_dec_div_3` is "No". If the value is "No", the scale is calculated as $31-p+s$. If set to "Yes", the scale is calculated as $\text{MAX}(3, 31-p+s)$. This causes the result of decimal division to always have a scale of at least 3. Precision is always 31.

Effects of changing the value of `min_dec_div_3`

Changing this database configuration parameter might cause changes to applications for existing databases. This behavior can occur when the resulting scale for decimal division would be impacted by changing this database configuration parameter. The following list shows some possible scenarios that might affect applications. Consider these scenarios before you change the `min_dec_div_3` configuration parameter on a database server with existing databases.

- A static package will not change behavior until the package is rebound, either implicitly or explicitly. For example, after changing the value from NO to YES, the additional scale digits might not be included in the results until rebind occurs. After changing the value of `min_dec_div_3`, recompile all static SQL packages whose results are effected by the change to force a rebind. You can force an explicit rebind by running the **REBIND** command or the **db2rbind** command.
- Materialized query tables (MQTs) might contain different results after you alter the `min_dec_div_3` configuration parameter. To ensure that previously created MQTs contain only data that adheres to the new format, refresh these MQTs by using the REFRESH TABLE statement.
- The results of a trigger might be affected by the changed format. Altering the `min_dec_div_3` value has no effect on data that has already been written.
- A check constraint that involves decimal division might restrict some values that were previously accepted. Such rows now violate the constraint but are not detected until one of the following events occurs:
 - One of the columns that are involved in the check constraint row is updated
 - The SET INTEGRITY statement with the IMMEDIATE CHECKED option is processed

To force checking of such a constraint, follow these steps:

1. Run the ALTER TABLE statement to drop the check constraint
 2. Run the ALTER TABLE statement to add the constraint
- After you change the value of `min_dec_div_3`, recompile all static SQL packages that depend on the value of a generated column whose results are effected by the change in the `min_dec_div_3` value. To find out which static SQL packages are effected, you must compile, rebind all the packages by running the **db2rbind** command.
 - An index with expression-based keys whose calculation is dependent on `min_dec_div_3` might be different for identical rows. The difference in values occurs if one row was inserted before the change to `min_dec_div_3` and the other was inserted after. Drop and recreate all potentially impacted expression-based indexes after you change the value of the `min_dec_div_3` configuration parameter. If you are unsure that a particular expression-based index is impacted, drop and recreate the index to avoid incorrect values in the index.

Note: `min_dec_div_3` also has the following limitations:

1. The command **GET DB CFG FOR DBNAME** will not display the **min_dec_div_3** setting. The best way to determine the current setting is to observe the side-effect of a decimal division result. For example, consider the following statement:

```
VALUES (DEC(1,31,0)/DEC(1,31,5))
```

If this statement returns sqlcode SQL0419N, the database does not have **min_dec_div_3** support, or it is set to "No". If the statement returns 1.000, **min_dec_div_3** is set to "Yes".

2. **min_dec_div_3** does not appear in the list of configuration keywords when you run the following command: ? UPDATE DB CFG

mincommit - Number of commits to group

This parameter allows you to delay the writing of log records to disk until a minimum number of commits have been performed, helping to reduce the processing time the database manager requires when writing log records.

Important: This parameter is deprecated in Version 10.1 and might be removed in a future release. This parameter can still be used in releases before Version 10.1. In Version 10.1 and later releases, the value specified for this configuration parameter is ignored.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

1 [1 - 25]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Counter

This delay might improve performance when you have multiple applications running against a database and many commits are requested by the applications within a very short time frame.

This grouping of commits will only occur when the value of this parameter is greater than one and when the number of applications connected to the database is greater than or equal to the value of this parameter. When commit grouping is being performed, application commit requests could be held until either one second has elapsed or the number of commit requests equals the value of this parameter.

This parameter should be incremented by small amounts only; for example one (1). You should also use multi-user tests to verify that increasing the value of this parameter provides the expected results. Setting this parameter too high can negatively impact application response time.

Changes to the value specified for this parameter take effect immediately; you do not have to wait until all applications disconnect from the database.

Recommendation: It is recommended that this parameter is set to the default of 1.

You could sample the number of transactions per second and adjust this parameter to accommodate the peak number of transactions per second (or some large percentage of it). Accommodating peak activity would minimize the processing time of writing log records during transaction intensive periods.

If you increase **mincommit**, you might also need to increase the **logbufsz** parameter to avoid having a full log buffer force a write during these transaction intensive periods. In this case, the **logbufsz** should be equal to:

`mincommit * (log space used, on average, by a transaction)`

You can use the database system monitor to help you tune this parameter in the following ways:

- Calculating the peak number of transactions per second:

Taking monitor samples throughout a typical day, you can determine your transaction intensive periods. You can calculate the total transactions by adding the following monitor elements:

- **commit_sql_stmts** (commit statements attempted)
- **rollback_sql_stmts** (rollback statements attempted)

Using this information and the available timestamps, you can calculate the number of transactions per second.

- Calculating the log space used per transaction:

Using sampling techniques over a period of time and a number of transactions, you can calculate an average of the log space used with the following monitor element:

- **log_space_used** (unit of work log space used)

mirrorlogpath - Mirror log path

This parameter specifies a string of up to 242 bytes for the mirror log path. The string must point to a fully qualified path name.

Configuration type

Database

Parameter type

Configurable

Default [range]

Null [any valid path or device]

In both Db2 pureScale environments and Db2 Enterprise Server Edition environments, the database partition number and a log stream ID are automatically appended to the path, for example, `/home/dbuser/dblogs/NODE0000/LOGSTREAM0000/`.

If you set the value of the **mirrorlogpath** configuration parameter, the Db2 database system creates active log files in both the log path and the mirror log path. All log data is written to both paths. The mirror log path has a duplicate set of active log files, such that if there is a disk error or human error that corrupts active log files on one of the paths, the database can still function.

In a Db2 pureScale environment, the first member connecting to or activating the database processes the changes to the value of this log path parameter. The Db2 database manager verifies that the path exists and that it has both read and write access to that path.

The database manager also attempts to create member-specific subdirectories for the log files. If any one of these operations fails, the Db2 database manager rejects the specified path and brings the database online using the old path. If the database manager accepts the specified path, the new value is propagated to each member. If a member fails while trying to switch to the new path, subsequent attempts to activate it or to connect to it fail with SQL5099N. All members must use the same log path.

If you change the mirror log path, there might be log files in the old mirror log path. These log files might not have been archived, so you might need to archive these log files manually. Also, if you are running replication on the related database, replication might still need the log files from before the log path change. If you configured the database to use log archiving and either the Db2 database system automatically archived all the log files or you archived them manually, the Db2 database system can retrieve the log files to complete the replication process. Otherwise, you can copy the files from the old mirror log path to the new mirror log path.

If the **logpath** or **newlogpath** configuration parameter specifies a raw device as the location where the log files are stored, mirror logging, as indicated by the **mirrorlogpath** configuration parameter, is not allowed. If the **logpath** or **newlogpath** configuration parameter specifies a file path as the location where the log files are stored, mirror logging is allowed, and the **mirrorlogpath** configuration parameter must specify a file path.

The **mirrorlogpath** configuration parameter affects log archiving behavior when you also set the **logarchmeth2** configuration parameter. When you set the values of both the **mirrorlogpath** and **logarchmeth2** configuration parameters, the **logarchmeth2** configuration parameter archives log files from the mirror log path instead of archiving additional copies of the log files in the active log path. You can use this log archiving behaviour to improve resilience during rollforward recovery, because a usable archived log file from the mirror log path might still be available to continue a database recovery operation even if a primary log file became corrupted before archiving.

Recommendation:

- Just like the log files, the mirror log files should be on a physical disk that does not have high I/O.
- It is strongly recommended that the mirror log path and the primary log path be on separate devices.

You can use the database system monitor to track the number of I/Os related to database logging. The following data elements return the amount of I/O activity related to database logging.

log_reads

The number of log pages read.

log_writes

The number of log pages written.

You can use an operating system monitor tool to collect information about other disk I/O activity, and then compare the two types of I/O activity.

mon_act_metrics - Monitoring activity metrics configuration parameter

This parameter controls the collection of activity metrics on the entire database and affects activities submitted by connections associated with any Db2 workload definitions.

Configuration type

Database

Parameter type

- Configurable online

Note: If you modify the **mon_act_metrics** parameter online, only the new activities use the updated value. The existing activities are not affected.

Default [range]

BASE [NONE, BASE, EXTENDED]

Upgrade Note

- On databases created before V9.7 and then upgraded to V9.7 or higher, the **mon_act_metrics** parameter is set to NONE by default.
- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **mon_req_metrics** is set to the value on member 0.

If you set this configuration parameter to BASE, activity metrics are collected for all activities executed on the data server regardless of the Db2 workload the connection that submitted the activity is associated with.

If you set this configuration parameter to EXTENDED, the same metrics are collected as under the BASE setting. In addition, the values reported for the following monitor elements are determined with more granularity:

- **total_section_time**
- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

For information about how the EXTENDED setting affects these monitor elements, see the detailed monitor element descriptions.

If you set this configuration parameter to NONE, activity metrics are collected only for the subset of activities submitted by a connection that is associated with a Db2 workload whose COLLECT ACTIVITY METRICS clause has been set to BASE

The following monitor interfaces report activity metrics:

- MON_GET_PKG_CACHE_STMT_DETAILS
- MON_GET_ACTIVITY
- MON_GET_ACTIVITY_DETAILS
- MON_GET_PKG_CACHE_STMT
- WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES

- Package cache event monitor
- Activity event monitor

If you use these interfaces, consider setting **mon_act_metrics** to a value other than NONE. Each interface topic contains a list of the monitor elements that are reported and the details on whether a monitor element is controlled by the **mon_act_metrics** database configuration parameter.

mon_deadlock - Monitoring deadlock configuration parameter

This parameter controls the generation of deadlock events at the database level for the lock event monitor.

Configuration type

Database

Parameter type

- Configurable online

Default [range]

WITHOUT_HIST [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

If you set this parameter to NONE, no deadlock events will be generated unless deadlock event collection is enabled on Db2 Workload objects using the COLLECT DEADLOCK DATA clause.

If you set the parameter to WITHOUT_HIST, the data about deadlock events are sent to any active locking event monitor when the deadlock event occurs. The data collected for the deadlock wait event will not include a history of activities executed by the application waiting on the deadlock.

If you set the parameter to HISTORY, the lock wait event will include a history of activities executed by the application waiting on the lock. The history only includes activities executed in the current transaction for the application and will only report the newest activities executed if the maximum sized is reached. The default history size is 250. The history size can be configured using the **DB2_MAX_INACT_STMTS** registry variable.

If you set the parameter value to HIST_AND_VALUES, activity history collected with the lock timeout event will include the input data values for those activities that have them. These data values will not include LOB data, LONG VARCHAR data, LONG VARGRAPHIC data, structured type data, or XML data. For SQL statements compiled using the **REOPT ALWAYS** bind option, there will be no REOPT compilation or statement execution data values provided in the event information.

This parameter controls the collection deadlock events at the database level for the lock event monitor. The **mon_deadlock** parameter determines whether or not a deadlock wait event will be collected by the lock event monitor when a deadlock occurs.

The **mon_deadlock** parameter value represents a minimum level of collection that is enabled for all Db2 applications. When individual Db2 workloads specify a higher level of collection than the configuration parameter, the Db2 workload setting is used instead of the configuration parameter value. You should note that system applications do not run in a workload and so the **mon_deadlock** parameter is the only way for system applications to collect deadlock data.

To capture the deadlocks with the lock event monitor, as lock waiters or lock holders might span workloads, enable the deadlock collection at the database level. The level of data collected by a deadlock can be controlled individually at the workload level or can be set at the database level by this parameter.

mon_locktimeout - Monitoring lock timeout configuration parameter

This parameter controls the generation of lock timeout events at the database level for the lock event monitor and affects all Db2 workload definitions.

Configuration type

Database

Parameter type

- Configurable online

Default [minimum level of collection that is enabled for all workloads or service classes on the database]

NONE [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

If you set this parameter to NONE, no lock timeout events will be generated unless lock timeout event collection is enabled on Db2 Workload objects using the COLLECT LOCK TIMEOUT DATA clause.

If you set the parameter to WITHOUT_HIST, the data about lock events are sent to any active locking event monitor when the lock event occurs. The data collected for the lock timeout event will not include a history of activities executed by the application waiting on the lock.

If you set the parameter to HISTORY, the lock timeout event will include a history of activities executed by the application waiting on the lock. The history only includes activities executed in the current transaction for the application and will only report the newest activities executed if the maximum sized is reached. The default history size is 250. The history size can be configured using the **DB2_MAX_INACT_STMTS** registry variable.

If you set the parameter value to HIST_AND_VALUES, activity history collected with the lock timeout event will include the input data values for those activities that have them. These data values will not include LOB data, LONG VARCHAR data, LONG VARGRAPHIC data, structured type data, or XML data. For SQL statements compiled using the **REOPT ALWAYS** bind option, there will be no REOPT compilation or statement execution data values provided in the event information.

This parameter controls the collection of lock timeout events at the database level for the lock event monitor. The **mon_locktimeout** parameter determines whether or not a lock timeout event will be collected by the lock event monitor when an application times out waiting on a lock.

The **mon_locktimeout** parameter value represents a minimum level of collection that is enabled for all Db2 applications. The collection of lock timeout events can be enabled for a subset of Db2 applications using the COLLECT LOCK TIMEOUT DATA clause on a Db2 Workload object instead of the **mon_locktimeout** parameter.

When individual Db2 workloads specify a higher level of collection than the configuration parameter, the Db2 workload setting is used instead of the configuration parameter value. You should note that system applications do not

run in a workload and so the **mon_locktimeout** parameter is the only way for system applications to collect lock timeout data.

mon_lockwait - Monitoring lock wait configuration parameter

This parameter controls the generation of lock wait events at the database level for the lock event monitor.

Configuration type

Database

Parameter type

- Configurable online

Default [range]

NONE [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

If you set this parameter to NONE, no lock wait events will be generated unless lock wait event collection is enabled on Db2 Workload objects using the COLLECT LOCK WAIT DATA clause.

If you set the parameter to WITHOUT_HIST, the data about lock events are sent to any active locking event monitor when the lock event occurs. The data collected for the lock wait event will not include a history of activities executed by the application waiting on the lock.

If you set the parameter to HISTORY, the lock wait event will include a history of activities executed by the application waiting on the lock. The history only includes activities executed in the current transaction for the application and will only report the newest activities executed if the maximum sized is reached. The default history size is 250. The history size can be configured using the **DB2_MAX_INACT_STMTS** registry variable.

If you set the parameter value to HIST_AND_VALUES, activity history collected with the lock timeout event will include the input data values for those activities that have them. These data values will not include LOB data, LONG VARCHAR data, LONG VARGRAPHIC data, structured type data, or XML data. For SQL statements compiled using the **REOPT ALWAYS** bind option, there will be no REOPT compilation or statement execution data values provided in the event information.

The **mon_lockwait** parameter value represents a minimum level of collection that is enabled for all Db2 applications. The collection of lock wait events can be enabled for a subset of Db2 applications using the COLLECT LOCK WAIT DATA clause on a Db2 Workload object instead of the **mon_lockwait** parameter.

When individual Db2 workloads specify a higher level of collection than the configuration parameter, the Db2 workload setting is used instead of the configuration parameter value. You should note that system applications do not run in a workload and so the **mon_lockwait** parameter is the only way for system applications to collect lock wait data.

This parameter controls the collection of lock wait events at the database level for the lock event monitor. The **mon_lockwait** configuration parameter is used in conjunction with the **mon_lw_thresh** configuration parameter. The **mon_lockwait** parameter determines whether or not a lock wait event will be collected by the lock event monitor when an application waits longer than **mon_lw_thresh** microseconds for a lock.

mon_lw_thresh - Monitoring lock wait threshold configuration parameter

This parameter controls the amount of time spent in lock wait before an event for **mon_lockwait** is generated.

Configuration type

Database

Parameter type

- Configurable online

Default [range]

5000000 [1000 ... MAX_INT]

Upgrade Note

On databases created before V9.7 and then upgraded to V9.7 or higher, the **mon_lw_thresh** parameter is set to 4294967295 by default.

Unit of measure

Microseconds

When this parameter is set both at the database level and at the workload level, the shorter of the two configured times is considered for the given workload.

mon_lck_msg_lvl - Monitoring lock event notification messages configuration parameter

This parameter controls the logging of messages to the administration notification log when lock timeout, deadlock, and lock escalation events occur.

Configuration type

Database

Parameter type

Configurable online

Default [range]

1 [0 - 3]

With the occurrence of lock timeout, deadlock, and lock escalation events, messages can be logged to the administration notification log by setting this database configuration parameter to a value appropriate for the level of notification that you want. The following list outlines the levels of notification that can be set:

- 0** Level 0: No notification of lock escalations, deadlocks, and lock timeouts is provided
- 1** Level 1: Notification of lock escalations
- 2** Level 2: Notification of lock escalations and deadlocks
- 3** Level 3: Notification of lock escalations, deadlocks, and lock timeouts

The default level of notification setting for this database configuration parameter is 1.

mon_obj_metrics - Monitoring object metrics configuration parameter

This parameter controls the collection of data object metrics on an entire database.

Configuration type

Database

Parameter type

- Configurable online

Default [range]

EXTENDED [NONE, BASE, EXTENDED]

Upgrade Note

- If you are upgrading from Version 9.8 or earlier, the **mon_obj_metrics** configuration parameter retains the value that it had before you upgraded your database.
- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **mon_req_metrics** is set to the value on member 0.

Set this configuration parameter to **BASE** to collect all metrics that are reported through the following table functions:

- MON_GET_BUFFERPOOL
- MON_GET_CONTAINER
- MON_GET_TABLESPACE

Set this configuration parameter to **EXTENDED** to collect additional metrics that are reported through the following table functions:

Table 13. Metrics returned with EXTENDED option

Table Function	Metrics Reported
MON_GET_INDEX	object_index_gbp_indep_pages_found_in_lbp object_index_gbp_invalid_pages object_index_gbp_l_reads object_index_gbp_p_reads object_index_l_reads object_index_lbp_pages_found object_index_p_reads
MON_GET_INDEX_USAGE_LIST	object_index_gbp_indep_pages_found_in_lbp object_index_gbp_invalid_pages object_index_gbp_l_reads object_index_gbp_p_reads object_index_l_reads object_index_lbp_pages_found object_index_p_reads

Table 13. Metrics returned with EXTENDED option (continued)

Table Function	Metrics Reported
MON_GET_TABLE	direct_read_reqs direct_reads direct_write_reqs direct_writes lock_escals lock_escals_global lock_wait_time lock_wait_time_global lock_waits lock_waits_global object_data_gbp_indep_pages_found_in_lbp object_data_gbp_invalid_pages object_data_gbp_l_reads object_data_gbp_p_reads object_data_l_reads object_data_lbp_pages_found object_data_p_reads object_xda_gbp_indep_pages_found_in_lbp object_xda_gbp_invalid_pages object_xda_gbp_l_reads object_xda_gbp_p_reads object_xda_l_reads object_xda_lbp_pages_found object_xda_p_reads
MON_GET_TABLE_USAGE_LIST	direct_read_reqs direct_reads direct_write_reqs direct_writes lock_escals lock_escals_global lock_wait_time lock_wait_time_global lock_waits lock_waits_global object_data_gbp_indep_pages_found_in_lbp object_data_gbp_invalid_pages object_data_gbp_l_reads object_data_gbp_p_reads object_data_l_reads object_data_lbp_pages_found object_data_p_reads object_xda_gbp_indep_pages_found_in_lbp object_xda_gbp_invalid_pages object_xda_gbp_l_reads object_xda_gbp_p_reads object_xda_l_reads object_xda_lbp_pages_found object_xda_p_reads overflow_accesses overflow_creates rows_deleted rows_inserted rows_read rows_updated

If you set this configuration parameter to NONE, the metrics reported through the previously mentioned table functions are not collected.

mon_pkglist_sz - Monitoring package list size configuration parameter

This parameter controls the maximum number of entries that can appear in the package listing per unit of work as captured by the unit of work monitor.

Configuration type

Database

Parameter type

Configurable online

Propagation clause

Next unit of work

Default [range]

32 [0 - 1024]

Unit of measure

Number of entries in the package list

The package list will have a maximum size as specified by the value for this database configuration parameter. The size of the package list is determined at the start of the unit of work. Changes to the package list size are not reflected until the following unit of work. The default size for the package list is 32 entries.

mon_req_metrics - Monitoring request metrics configuration parameter

This parameter controls the collection of request metrics on the entire database and affects requests executing in any Db2 service classes.

Configuration type

Database

Parameter type

- Configurable online

Default [range]

BASE [NONE, BASE, EXTENDED]

Upgrade Note

- On databases created before V9.7 and then upgraded to V9.7 or higher, the **mon_req_metrics** parameter is set to NONE by default.
- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **mon_req_metrics** is set to the value on member 0.

If you set this configuration parameter to BASE, request metrics are collected for all requests executed on the data server, irrespective of the Db2 service class the request runs in.

If you set this configuration parameter to EXTENDED, the same metrics are collected as under the BASE setting. In addition, the values reported for the following monitor elements are determined with more granularity:

- **total_section_time**
- **total_section_proc_time**

- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

For information about how the EXTENDED setting affects these monitor elements, refer to the detailed monitor element descriptions.

If you set this configuration parameter to NONE, request metrics are collected only for the subset of requests running in a Db2 service class whose service superclass has the COLLECT REQUEST METRICS clause set to BASE.

The following monitor interfaces report request metrics:

- MON_GET_UNIT_OF_WORK
- MON_GET_UNIT_OF_WORK_DETAILS
- MON_GET_CONNECTION
- MON_GET_CONNECTION_DETAILS
- MON_GET_DATABASE
- MON_GET_DATABASE_DETAILS
- MON_GET_ROUTINE
- MON_GET_ROUTINE_DETAILS
- MON_GET_SERVICE_SUBCLASS
- MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_WORKLOAD
- MON_GET_WORKLOAD_DETAILS
- Statistics event monitor (DETAILS_XML monitor element in the event_wlstats and event_scstats logical data groups)
- Unit of work event monitor

If you use these interfaces, consider setting **mon_req_metrics** to a value other than NONE. Each interface topic contains a list of the monitor elements that are reported and the details on whether a monitor element is controlled by the **mon_req_metrics** database configuration parameter.

mon_rtn_data - Monitoring routine capture

The **mon_rtn_data** configuration parameter controls the capture of routine invocations. It is a parent parameter to the **mon_rtn_excllist** configuration parameter.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Next routine invocation

Default [range]

NONE [NONE,BASE]

If the value of `mon_rtn_data` is set to `BASE`, then routine monitoring is enabled. Information about executed routines can be accessed using the `MON_GET_ROUTINE` and `MON_GET_ROUTINE_DETAILS` interfaces. Routine information is stored in the database heap memory pool. Routine statement monitoring can also be enabled using the `mon_rtn_execlist` configuration parameter when `mon_rtn_data` is set to `BASE`.

If the value of `mon_rtn_data` is set to `NONE`, then routine monitoring is disabled. No information is reported in the `MON_GET_ROUTINE`, `MON_GET_ROUTINE_DETAILS`, and `MON_GET_ROUTINE_EXEC_LIST` interfaces. When the `mon_rtn_data` value is changed to `NONE`, all preexisting routine monitoring information is deleted.

mon_rtn_execlist - Monitoring routine executable list

This parameter controls the monitoring of statements executed by routines. This is done at the database level and takes effect for any routine invocation within the database.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Next routine invocation

Default [range]

OFF [OFF, ON]

Routine executable lists are a list of sections (compiled SQL statements) executed by any procedures, external functions, compiled functions, compiled triggers, and anonymous blocks. Routine executable lists include a set of metrics for each section aggregated over all the executions of the section by a routine. Each section in the output executable lists is uniquely identified by an executable ID which can be used to find additional information about the sections, such as statement text, with package cache interfaces like `MON_GET_PKG_CACHE_STMT`. Memory for the routine executable lists file is allocated from the monitor heap. Routine executable list monitoring is limited to at most 20% of the available monitor heap.

When the value of `mon_rtn_execlist` is set to `ON`, then executable ID list collection is enabled for routine monitoring, and `MON_GET_RTN_EXEC_LIST` returns monitoring data.

When the value of `mon_rtn_execlist` is set to `OFF`, then executable ID list collection is disabled for routine monitoring, and `MON_GET_RTN_EXEC_LIST` does not return monitoring data. When the value of `mon_rtn_execlist` is changed to `OFF`, all preexisting routine execution list metrics is deleted.

For a list of metrics gathered when this parameter is enabled, refer to the `MON_GET_RTN_EXEC_LIST` documentation.

mon_uow_data - Monitoring unit of work events configuration parameter

This parameter specifies whether information about a unit of work, also referred to as a transaction, is sent to the active unit of work event monitors when the unit of work is completed. It is a parent parameter to the **mon_uow_execlist** and **mon_uow_pkglist** configuration parameters.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Next unit of work

Default [range]

NONE [NONE, BASE]

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **mon_uow_data** is set to the value on member 0.

If you disable this parameter, all of its child parameters are also disabled, but their settings, as recorded in the database configuration file, do not change. If you enable this parent parameter, recorded values for its child parameters take effect. This means that data collection can be enabled or disabled globally.

If the parameter is set to BASE, basic data collection for a UOW event monitor is enabled. Information about all units of work that are executed on the data server are sent to the active unit of work event monitors when the units of work are completed.

If the parameter is set to NONE, all data collection for a UOW event monitor is disabled. Information is sent to the unit of work event monitors only for those units of work that are executed under a Db2 workload whose COLLECT UNIT OF WORK DATA clause is set to BASE. The default setting for the **mon_uow_data** configuration parameter is NONE.

The information that is gathered at the end of a unit of work includes the system-level request metrics for that unit of work, for example, the amount of CPU that is used during the unit of work. The collection of these request metrics is controlled independently from the collection of the unit of work data. You collect unit of work data by using either the COLLECT REQUEST METRICS clause with a Db2 service superclass or by setting the **mon_req_metrics** database configuration parameter to the value of BASE. If you do not enable request metrics collection, the value of all the request metrics that are gathered as part of the unit of work data is 0.

mon_uow_execlist - Monitoring unit of work events with executable list configuration parameter

This parameter controls the generation of unit of work events with executable ID listing information included. This is done at the database level for the unit of work event monitor. The **mon_uow_execlist** database configuration parameter is a child parameter of the **mon_uow_data** database configuration parameter.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Next unit of work

Default [range]

OFF [OFF, ON]

Valid values for this parameter are as follows:

- OFF** Executable ID list collection is disabled for a unit of work event monitor.
- ON** Executable ID list collection is enabled for a unit of work event monitor.

mon_uow_pkglist - Monitoring unit of work events with package list configuration parameter

This parameter controls the generation of unit of work events with package listing information included. This is done at the database level for the unit of work event monitor. The **mon_uow_pkglist** database configuration parameter is a child parameter of the **mon_uow_data** database configuration parameter.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Next unit of work

Default [range]

OFF [OFF, ON]

Valid values for this parameter are as follows:

- OFF** Package list collection is disabled for a unit of work event monitor.
- ON** Package list collection is enabled for a unit of work event monitor.

multipage_alloc - Multipage file allocation enabled

Multipage file allocation is used to improve insert performance. It applies to SMS table spaces only. If enabled, all SMS table spaces are affected: there is no selection possible for individual SMS table spaces.

Configuration type

Database

Parameter type

Informational

The default for the parameter is Yes: multipage file allocation is enabled.

Following database creation, this parameter cannot be set to No. Multipage file allocation cannot be disabled once it has been enabled. The **db2empfa** tool can be used to enable multipage file allocation for a database that currently has it disabled.

nchar_mapping - National character mapping

This parameter determines the data type mapping for national character string data types in Unicode databases.

Configuration type

Database

Parameter type

Configurable online

Default [range]

CHAR_CU32 [CHAR_CU32, GRAPHIC_CU32, GRAPHIC_CU16, NOT APPLICABLE]

If you are not using a Unicode database, the value of **nchar_mapping** is NOT APPLICABLE because there is no support for specifying national character string data types in non-Unicode databases.

The national character string data type mapping is as follows:

Table 14. Mapping between national character strings and data types

Parameter Value	NCHAR	NVARCHAR	NCLOB
CHAR_CU32	CHAR in string units CODEUNITS32	VARCHAR in string units CODEUNITS32	CLOB in string units CODEUNITS32
GRAPHIC_CU32	GRAPHIC in string units CODEUNITS32	VARGRAPHIC in string units CODEUNITS32	DBCLOB in string units CODEUNITS32
GRAPHIC_CU16	GRAPHIC in string units CODEUNITS16	VARGRAPHIC in string units CODEUNITS16	DBCLOB in string units CODEUNITS16

Effects of changing the value of nchar_mapping

- Materialized query tables (MQTs) might contain different results after altering the **nchar_mapping** value. To ensure that previously created MQTs contain only data that adheres to the new format, refresh these MQTs by using the REFRESH TABLE statement.
- The results of a trigger might be affected by the changed format. Altering the **nchar_mapping** value has no effect on data that has already been written.

- Constraints that allowed data to be inserted into a table might, if reevaluated, reject that same data. Similarly, constraints that did not allow data to be inserted into a table might, if reevaluated, accept that same data. Use the SET INTEGRITY statement to check for and correct data in a table that might no longer satisfy a constraint.
- After changing the value of **nchar_mapping**, recompile all static SQL packages that depend on the value of a generated column whose results are effected by the change in the **nchar_mapping** value. To find out which static SQL packages are effected, you must compile, rebind all the packages using the **db2rbind** command.
- The value of an index with expression-based keys whose calculation is dependent on **nchar_mapping** will be different after changing the value of **nchar_mapping**. Drop and recreate all potentially impacted expression-based indexes after changing the value of **nchar_mapping**. If you are not sure that a particular expression-based index is impacted, it is best to drop and recreate the index to avoid incorrect values in the index.

newlogpath - Change the database log path

This parameter allows you to specify a string of up to 242 bytes to change the location where the log files are stored.

Configuration type

Database

Parameter type

Configurable

Default [range]

Null [any valid path or device]

Important: The use of raw devices for database logging has been discontinued starting in Db2 Version 11.1 and will not be available in future releases.

If the string points to a path name, it must be a fully qualified path name, not a relative path name.

In both Db2 pureScale environments and Db2 Enterprise Server Edition environments, the database partition number and a log stream ID are automatically appended to the path, for example, /home/dbuser/db1ogs/NODE0000/LOGSTREAM0000/.

In a Db2 pureScale environment, the first member connecting to or activating the database processes configuration changes to this log path parameter. The Db2 database manager verifies that the path exists and that it has both read and write access to that path. It also creates member-specific subdirectories for the log files.

If any one of these operations fails, the Db2 database manager rejects the specified path and brings the database online using the old path. If the specified path is accepted, the new value is propagated to each member. If a member fails while trying to switch to the new path, subsequent attempts to activate it or to connect to it will fail (SQL5099N). All members must use the same log path.

The new setting does not become the value of **logpath** until both of the following events occur:

- The database is in a consistent state, as indicated by the **database_consistent** parameter.

- All applications are disconnected from the database

When the first new connection is made to the database, the database manager will move the logs to the new location specified by **logpath**.

There might be log files in the old log path. These log files might not have been archived. You might need to archive these log files manually. Also, if you are running replication on this database, replication might still need the log files from before the log path change. If the database is configured to use log archiving and if all the log files have been archived either by the Db2 database system automatically or by yourself manually, then the Db2 database system will be able to retrieve the log files to complete the replication process. Otherwise, you can copy the files from the old log path to the new log path.

If **logpath** or **newlogpath** specifies a file path as the location where the log files are stored, mirror logging is allowed and **mirrorlogpath** must also specify a file path.

Recommendation: Ideally, the log files will be on a physical disk which does **not** have high I/O. For instance, avoid putting the logs on the same disk as the operating system or high volume databases. This will allow for efficient logging activity and will reduce additional processing time taken, such as when waiting for I/O.

You can use the database system monitor to track the number of I/Os related to database logging.

The monitor elements **log_reads** (number of log pages read) and **log_writes** (number of log pages written) return the amount of I/O activity related to database logging. You can use an operating system monitor tool to collect information about other disk I/O activity, then compare the two types of I/O activity.

Do not use a network or local file system that is shared as the log path for both the primary and standby databases in a Db2 High Availability Disaster Recovery (HADR) database pair. The primary and standby databases each have copies of the transaction logs - the primary database ships logs to the standby database. If the log path for both the primary and standby databases points to the same physical location, then the primary and standby database would use the same physical files for their corresponding copies of the logs. The database manager returns an error if the database manager detects a shared log path.

num_db_backups - Number of database backups

This parameter specifies the number of full database backups to retain for a database.

Configuration type

Database

Parameter type

Configurable online

Propagation class

Transaction boundary

Default [range]

12 [1 - 32 767]

After the specified number of full database backups is reached, old backups are marked as expired in the recovery history file. Recovery history file entries for the table space backups and load copy backups that are related to the expired database backup are also marked as expired. When a backup is marked as expired, the physical backups can be removed from where they are stored (for example, disk, tape, Tivoli Storage Manager). The next database backup prunes the expired entries from the recovery history file.

The **rec_his_retentn** configuration parameter should be set to a value compatible with the value of **num_db_backups**. For example, if **num_db_backups** is set to a large value, the value for **rec_his_retentn** should be large enough to support the number of backups set as **num_db_backups**.

If both the **num_db_backups** and **rec_his_retentn** configuration parameters are set, backups are not removed unless both the **num_db_backups** and **rec_his_retentn** conditions are satisfied.

num_freqvalues - Number of frequent values retained

This parameter allows you to specify the number of “most frequent values” that will be collected when the **WITH DISTRIBUTION** option is specified on the **RUNSTATS** command.

Configuration type

Database

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

10 [0 - 32 767]

Unit of measure

Counter

Increasing the value of this parameter increases the amount of statistics heap (**stat_heap_sz**) used when collecting statistics.

The “most frequent value” statistics help the optimizer understand the distribution of data values within a column. A higher value results in more information being available to the query optimizer but requires additional catalog space. When 0 is specified, no frequent-value statistics are retained, even if you request that distribution statistics be collected.

You can also specify the number of frequent values retained as part of the **RUNSTATS** command at the table or the column level by using the **NUM_FREQVALUES** command parameter. If none is specified, the **num_freqvalues** configuration parameter value is used. Changing the number of frequent values retained through the **RUNSTATS** command is easier than making the change using the **num_freqvalues** database configuration parameter.

Updating this parameter can help the optimizer obtain better selectivity estimates for some predicates (=, <, >) over data that is non-uniformly distributed. More accurate selectivity calculations might result in the choice of more efficient access plans.

After changing the value of this parameter, you need to:

- Run the **RUNSTATS** command again to collect statistics with the changed number of frequent values
- Rebind any packages containing static SQL or XQuery statements.

When using **RUNSTATS**, you have the ability to limit the number of frequent values collected at both the table level and the column level. This allows you to optimize on space occupied in the catalogs by reducing the distribution statistics for columns where they could not be exploited and yet still using the information for critical columns.

Recommendation: In order to update this parameter you should determine the degree of non-uniformity in the most important columns (in the most important tables) that typically have selection predicates. This can be done using an SQL SELECT statement that provides an ordered ranking of the number of occurrences of each value in a column. You should not consider uniformly distributed, unique, long, or LOB columns. A reasonable practical value for this parameter lies in the range of 10 to 100.

Note that the process of collecting frequent value statistics requires significant CPU and memory (**stat_heap_sz**) resources.

num_iocleaners - Number of asynchronous page cleaners

This parameter allows you to specify the number of asynchronous page cleaners for a database.

Configuration type

Database

Parameter type

- Configurable
- Configurable by member in a Db2 pureScale environment

Default [range]

Automatic [0 - 255]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Counter

The number of castout engines is equal to the value that is set for the **num_iocleaners** configuration parameter. However, the maximum number of castout engines is 128.

These page cleaners write changed pages from the buffer pool to disk before the space in the buffer pool is required by a database agent. As a result, database agents should not have to wait for changed pages to be written out so that they might use the space in the buffer pool. This improves overall performance of the database applications.

The page cleaners will also decrease recovery time from soft failures, such as power outages, because the contents of the database on disk will be more up-to-date at any given time.

If this parameter is set to `AUTOMATIC`, the number of page cleaners started will be based on the number of physical CPU cores configured on the current machine, as well as the number of local logical database partitions in a partitioned database environment. There will always be at least one page cleaner started when this parameter is set to `AUTOMATIC`.

The number of page cleaners to start when this parameter is set to `AUTOMATIC` will be calculated using the following formula:

$$\text{number of page cleaners} = \max(\text{ceil}(\# \text{ CPUs} / \# \text{ local logical DPs}) - 1, 1)$$

This formula ensures that the number of page cleaners is distributed almost evenly across your logical database partitions, and that there are no more page cleaners than there are physical CPU cores.

Note: On the HP-UX platform, you must use the number of logical CPUs in the calculation, instead of the number of physical CPU cores.

Recommendation: Consider the following factors when setting the value for this parameter:

- Workload

Environments with high update transaction rates might require more page cleaners to be configured. This is only applicable when

DB2_USE_ALTERNATE_PAGE_CLEANING is `OFF`, which is also the default value.

- Buffer pool sizes

Environments with large buffer pools might also require more page cleaners to

be configured. This is only applicable when **DB2_USE_ALTERNATE_PAGE_CLEANING** is `OFF`, which is also the default value.

You can use the database system monitor to help you tune this configuration parameter using information from the event monitor about write activity from a buffer pool:

- The parameter can be reduced if both of the following conditions are true:

- **pool_data_writes** is approximately equal to **pool_async_data_writes**
- **pool_index_writes** is approximately equal to **pool_async_index_writes**.

Note: Reducing to a lower value, than what is computed via `AUTOMATIC`, is not recommended.

- The parameter should be increased if either of the following conditions are true:

- **pool_data_writes** is much greater than **pool_async_data_writes**
- **pool_index_writes** is much greater than **pool_async_index_writes**.

num_ioservers - Number of I/O servers

This parameter specifies the number of I/O servers for a database. No more than this number of I/Os for prefetching and utilities can be in progress for a database at any time.

Configuration type

Database

Parameter type

- Configurable
- Configurable by member in a Db2 pureScale environment

Default [range]

Automatic [1 - 255]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Counter

When allocated

When an application connects to a database

When freed

When an application disconnects from a database

I/O servers, also called prefetchers, are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities such as backup and restore. An I/O server waits while an I/O operation that it initiated is in progress. Non-prefetch I/Os are scheduled directly from the database agents and as a result are not constrained by **num_ioservers**.

When this parameter is set to AUTOMATIC, the number of prefetchers that are started is based on the parallelism settings of the table spaces in the current database partition. Parallelism settings are controlled by the **DB2_PARALLEL_IO** system environment variable. The number of prefetchers to start is calculated at database activation time that is based on the following formula:

$$\text{number of prefetchers} = \max(\max(\max \text{ over all table spaces}(\text{parallelism setting}), \text{number of cores} * \text{number of SMT threads}), 16)$$

Recommendation: In order to fully use all the I/O devices in the system, a good value to use is generally one or two more than the number of physical devices on which the database resides. It is better to configure more I/O servers, since there is some additional processing time that is associated with each I/O server and any unused I/O servers remain idle.

num_log_span - Number log span

This parameter specifies whether there is a limit to how many log files one transaction can span, and what that limit is.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

0 [0 - 65 535]

Unit of measure

Counter

If the value is not 0, this parameter indicates the number of active log files that one active transaction is allowed to span.

If the value is set to 0, there is no limit to how many log files one single transaction can span.

If an application violates the **num_log_span** configuration, the application is forced to disconnect from the database and the transaction is rolled back.

This parameter, along with the **max_log** configuration parameter, can be useful when enabling infinite active logspace. If infinite logging is on (that is, if *logsecond* is set to -1) then transactions are not restricted to the upper limit of the number of log files (*logprimary* + *logsecond*). When the value of *logprimary* is reached, Db2 starts to archive the active logs, rather than failing the transaction. This can cause problems if, for example, an application contains a long running transaction that is left uncommitted. If this occurs, the active logspace continues to grow, possibly leading to poor crash recovery performance. To prevent this, you can specify values for one or both of the **max_log** or **num_log_span** configuration parameters.

Note: The following Db2 commands are excluded from the limitation imposed by the **num_log_span** configuration parameter: ARCHIVE LOG, BACKUP DATABASE, LOAD, REORG, RESTORE DATABASE, and ROLLFORWARD DATABASE.

num_quantiles - Number of quantiles for columns

This parameter controls the number of quantiles that will be collected when the **WITH DISTRIBUTION** option is specified on the **RUNSTATS** command.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

20 [0 - 32 767]

Unit of measure

Counter

Increasing the value of this parameter increases the amount of statistics heap (**stat_heap_sz**) used when collecting statistics.

The “quantile” statistics help the optimizer understand the distribution of data values within a column. A higher value results in more information being available to the query optimizer but requires additional catalog space. When 0 or 1 is specified, no quantile statistics are retained, even if you request that distribution statistics be collected.

You can also specify the number of quantiles collected as part of the **RUNSTATS** command at the table or the column level, by using the **NUM_QUANTILES** command parameter. If none is specified, the **num_quantiles** configuration parameter value is used. Changing the number of quantiles that will be collected through the **RUNSTATS** command is easier than making the change using the **num_quantiles** database configuration parameter.

Updating this parameter can help obtain better selectivity estimates for range predicates over data that is non-uniformly distributed. Among other optimizer decisions, this information has a strong influence on whether an index scan or a table scan will be chosen. (It is more efficient to use a table scan to access a range of values that occur frequently and it is more efficient to use an index scan for a range of values that occur infrequently.)

After changing the value of this parameter, you need to:

- Run the **RUNSTATS** command again to collect statistics with the changed number of frequent values
- Rebind any packages containing static SQL or XQuery statements.

When using **RUNSTATS**, you have the ability to limit the number of quantiles collected at both the table level and the column level. This allows you to optimize on space occupied in the catalogs by reducing the distribution statistics for columns where they could not be exploited and yet still using the information for critical columns.

Recommendation: The default value for this parameter provides reasonably accurate estimates in most cases. You can consider increasing the value if you observe significant and consistent differences between:

- Selectivity estimates from the explain output; and
- Actual selectivity of range predicates over non-uniformly distributed column data.

A reasonable practical value for this parameter lies in the range of 10 to 50.

numarchretry - Number of retries on error

This parameter specifies the number of attempts that Db2 must make to archive a log file to the primary or the secondary archive directory before trying to archive log files to the failover directory.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

- Configurable online

Default [range]

5 [0 - 65 535]

This parameter is only used if the **failarchpath** database configuration parameter is set. If **failarchpath** parameter is not set, then the **numarchretry** configuration parameter is ignored and the database continuously retries archiving to the primary or the secondary log path.

numsegs - Default number of SMS containers

This parameter has been deprecated since Version 9.5, but is still being used by pre-Version 9.5 data servers and clients. Any value specified for this configuration parameter will be ignored by the database manager in Db2 Version 9.5 or later releases.

Note: The following information applies only to pre-Version 9.5 data servers and clients.

Configuration type

Database

Parameter type

Informational

Unit of measure

Counter

This parameter indicates the number of containers that will be created within the default table spaces. It also shows the information used when you created your database, whether it was specified explicitly or implicitly on the **CREATE DATABASE** command.

This parameter only applies to SMS table spaces; the **CREATE TABLESPACE** statement **does not** use it in any way.

number_compat - Number compatibility database configuration parameter

This parameter indicates whether the compatibility semantics associated with the NUMBER data type are applied to the connected database.

Configuration type

Database

Parameter type

Informational

The value is determined at database creation time, and is based on the setting of the **DB2_COMPATIBILITY_VECTOR** registry variable for NUMBER support. The value cannot be changed.

opt_direct_wrkld - Optimize directed workload configuration parameter

This database configuration parameter enables or disables explicit hierarchical locking (EHL). It affects the entire Db2 pureScale instance.

Configuration type

Database

Applies to

- Tables
- Range partition tables
- Partition indexes

Does not apply to

- System catalog tables

- Tables which are normally not shared across members, such as, temporary tables

Parameter type

Configurable Online

Default [range]

OFF [OFF, ON, YES, NO, AUTOMATIC]

The **opt_direct_wrkld** parameter specifies whether tables or range partitions are eligible to enter the NOT_SHARED state. It applies globally across all Db2 pureScale members. This parameter is supported only for instances that are part of Db2 pureScale environments.

This parameter does not require optimization profile support.

Usage notes

- If **opt_direct_wrkld** parameter is set to use AUTOMATIC, then EHL is enabled when both **CF_GBP_SZ** and **CF_LOCK_SZ** are also set to AUTOMATIC.
- Sample usage:
db2 update db cfg for <dbname> opt_direct_wrkld YES
- If the value of this parameter is changed from ON to OFF while the database is online, then tables cannot change to the NOT_SHARED state. Any tables that are already in NOT_SHARED state remain in this state until another member accesses the table.

overflowlogpath - Overflow log path

The **overflowlogpath** parameter specifies a location for Db2 databases to find log files needed for a rollforward operation, as well as where to store active log files retrieved from the archive. It also gives a location for finding and storing log files needed for using db2ReadLog API.

Configuration type

Database

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

NULL [any valid path]

This parameter can be used for several functions, depending on your logging requirements.

- Use the **overflowlogpath** parameter to specify a location for Db2 databases to find log files that are needed for a rollforward operation. It is similar to the **OVERFLOW LOG PATH** option on the **ROLLFORWARD** command. Instead of always specifying **OVERFLOW LOG PATH** on every **ROLLFORWARD** command, you can set this configuration parameter once. However, if both are used, the **OVERFLOW LOG PATH** option overwrites the **overflowlogpath** configuration parameter, for that particular rollforward operation.
- If **logsecond** is set to -1, you can specify a directory for Db2 to store active log files retrieved from the archive with the **overflowlogpath** parameter. Active log files have to be retrieved for rollback operations if they are no longer in the active log path. Without the **overflowlogpath** parameter set, Db2 databases retrieve the log files into the active log path. Use the **overflowlogpath** parameter

to provide additional resource for Db2 databases to store the retrieved log files. The benefit includes spreading the I/O cost to different disks, and allowing more log files to be stored in the active log path.

- If you need to use the db2ReadLog API for replication, for example, use the **overflowlogpath** to specify a location for Db2 databases to search for log files that are needed for this API. Before Db2 Version 8, db2ReadLog was called sqlurlog. If the log file is not found (in either the active log path or the overflow log path) and the database is configured to archive logs by using the **logarchmeth1** or **logarchmeth2** parameters, Db2 retrieves the log file. Also use the **overflowlogpath** parameter to specify a directory for Db2 databases to store the log files retrieved. The benefit comes from reducing the I/O cost on the active log path and allowing more log files to be stored in the active log path.
- You can specify a location for Db2 databases to retrieve log files that are required for a **BACKUP DATABASE INCLUDE LOGS** operation.

To set **overflowlogpath**, specify a string of up to 242 bytes. The string must point to a path name, and it must be a fully qualified path name, not a relative path name. The path name must be a directory, not a raw device. You cannot specify **overflowlogpath** as an empty string.

Note: The database partition number and a log stream ID are automatically appended to the path, for example, /home/dbuser/dblogs/NODE0000/LOGSTREAM0000/. If you are creating logs in a multiple partition database environment, such as Db2 pureScale or a partitioned database environment, you must copy log files into the log directories, otherwise your logs might be incomplete.

page_age_trgt_gcr - Page age target group crash recovery configuration parameter

This configuration parameter specifies the target duration (in seconds) for changed pages to be kept in the group buffer pool before the pages are persisted to disk or the caching facility. This parameter applies only to Db2 pureScale instances.

Configuration type

Database

Parameter type

Configurable Offline

Default [range]

240 [1 - 65535]

The **page_age_trgt_gcr** parameter must be greater than or equal to **page_age_trgt_mcr** database configuration parameter.

The **page_age_trgt_gcr** parameter is used when the **softmax** parameter is configured to a value of 0. Migrated databases retain the previous value for **softmax** and ignore the **page_age_trgt_gcr** parameter if this value is not 0. The value of **softmax** is set to 0 in new databases.

Increasing the value of this configuration parameter keeps changed pages in memory for a longer time, allowing more page updates to be buffered before the pages are persisted to disk. This behavior can help to improve performance but also increases recovery time.

page_age_trgt_mcr - Page age target member crash recovery configuration parameter

This configuration parameter specifies the target duration (in seconds) for changed pages to be kept in the local buffer pool before they are persisted to table space storage, or for Db2 pureScale instances, to table space storage or to the group buffer pool.

Configuration type

Database

Parameter type

Configurable Offline

Default [range]

240 [1 - 65535]

In a Db2 pureScale environment, the default is:

120 [1 - 65535]

The **page_age_trgt_mcr** parameter is used when the **softmax** parameter is configured to a value of 0. Migrated databases retain the previous value for **softmax** and ignore the **page_age_trgt_mcr** parameter if this value is not 0. The value of **softmax** is set to 0 in new databases. When the **softmax** parameter is set to 0 the **page_age_trgt_mcr** parameter is used to determine the frequency of soft checkpoints.

Increasing the value of this configuration parameter keeps changed pages in memory for a longer time, allowing more page updates to be buffered before the pages are persisted to disk. This behavior can help to improve performance but also increases recovery time.

pagesize - Database default page size

This parameter contains the value that was used as the default page size when the database was created. Possible values are: 4 096, 8 192, 16 384 and 32 768. When a buffer pool or table space is created in that database, the same default page size applies.

Configuration type

Database

Parameter type

Informational

pckcachesz - Package cache size

This parameter is allocated out of the database shared memory, and is used for caching of sections for static and dynamic SQL and XQuery statements on a database.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

32-bit operating systems

Automatic [-1, 32 - 128 000]

64-bit operating systems

Automatic [-1, 32 - 2 147 483 646]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

When the database is initialized

When freed

When the database is shut down

In a partitioned database system, there is one package cache for each database partition.

Caching packages allows the database manager to reduce its internal processing time by eliminating the need to access the system catalogs when reloading a package; or, in the case of dynamic SQL or XQuery statements, eliminating the need for compilation. Sections are kept in the package cache until one of the following events occurs:

- The database is shut down
- The package or dynamic SQL or XQuery statement is invalidated
- The cache runs out of space.

This caching of the section for a static or dynamic SQL or XQuery statement can improve performance, especially when the same statement is used multiple times by applications connected to a database. This is particularly important in a transaction processing environment.

When this parameter is set to `AUTOMATIC`, it is enabled for self tuning. When `self_tuning_mem` is set to `ON`, the memory tuner will dynamically size the memory area controlled by `pckcachesz` as the workload requirements change. Because the memory tuner trades memory resources between different memory consumers, there must be at least two memory consumers enabled for self tuning in order for self tuning to be active.

Automatic tuning of this configuration parameter will only occur when self tuning memory is enabled for the database (the `self_tuning_mem` configuration parameter is set to `ON`.)

When this parameter is set to `-1`, the value used to calculate the page allocation is eight times the value specified for the `maxappls` configuration parameter. The exception to this occurs if eight times `maxappls` is less than 32. In this situation, the default value of `-1` will set `pckcachesz` to 32.

Recommendation: When tuning this parameter, you should consider whether the extra memory being reserved for the package cache might be more effective if it was allocated for another purpose, such as the buffer pool or catalog cache. For this reason, you should use benchmarking techniques when tuning this parameter.

Tuning this parameter is particularly important when several sections are used initially and then only a few are run repeatedly. If the cache is too large, memory is wasted holding copies of the initial sections.

The following monitor elements can help you determine whether you should adjust this configuration parameter:

- **pkg_cache_lookups** (package cache lookups)
- **pkg_cache_inserts** (package cache inserts)
- **pkg_cache_size_top** (package cache high water mark)
- **pkg_cache_num_overflows** (package cache overflows)

Note: The package cache is a working cache, so you cannot set this parameter to zero. There must be sufficient memory allocated in this cache to hold all sections of the SQL or XQuery statements currently being executed. If there is more space allocated than currently needed, then sections are cached. These sections can simply be executed the next time they are needed without having to load or compile them.

The limit specified by the **pckcachesz** parameter is a soft limit. This limit can be exceeded, if required, if memory is still available in the database shared set. You can use the **pkg_cache_size_top** monitor element to determine the largest that the package cache has grown, and the **pkg_cache_num_overflows** monitor element to determine how many times the limit specified by the **pckcachesz** parameter has been exceeded.

pl_stack_trace

Starting in Db2 Version 10.5 Fix Pack 7, this parameter determines whether error stack logging is enabled or disabled for SQL PL and PL/SQL routines.

Configuration type

Database

Parameter type

Configurable online

Default [range]

NONE [NONE; ALL; UNHANDLED]

When this SQL PL and PL/SQL error stack logging configuration parameter is set to **NONE**, error stack logging is disabled; no error stack trace records will be written to the `db2diag.log` file.

When this parameter is set to **ALL**, error stack logging is enabled. Error stack trace records will be written to the `db2diag.log` file each time a new SQL error is detected within an SQL PL or PL/SQL routine.

When this configuration parameter is set to **UNHANDLED**, error stack logging is enabled. Error stack trace records will be written to the `db2diag.log` file each time a new SQL error is detected within an SQL PL or PL/SQL routine, but only if the error is not captured by an SQL PL condition handler or PL/SQL exception handler within the SQL routine that is running currently.

priv_mem_thresh - Private memory threshold

This parameter has been deprecated since Version 9.5, but is still being used by pre-Version 9.5 data servers and clients. Any value specified for this configuration parameter will be ignored by the database manager in Db2 Version 9.5 or later releases.

Note: The following information applies only to pre-Version 9.5 data servers and clients.

Configuration type

Database manager

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable

Default [range]

20 000 [-1; 32 - 112 000]

Unit of measure

Pages (4 KB)

This parameter is used to determine the amount of unused agent private memory that will be kept allocated, ready to be used by new agents that are started. It does not apply to Linux and UNIX platforms.

A value of -1 will cause this parameter to use the value of the `min_priv_mem` parameter.

Recommendation: When setting this parameter, you should consider the client connection/disconnection patterns as well as the memory requirements of other processes on the same machine.

If there is only a brief period during which many clients are concurrently connected to the database, a high threshold will prevent unused memory from being decommitted and made available to other processes. This case results in poor memory management which can affect other processes which require memory.

If the number of concurrent clients is more uniform and there are frequent fluctuations in this number, a high threshold will help to ensure memory is available for the client processes and can reduce the processing time taken to allocate and deallocate memory.

rec_his_retentn - Recovery history retention period

This parameter specifies the number of days that historical information on backups are retained.

Configuration type

Database

Parameter type

Configurable online

Propagation class

Transaction boundary

Default [range]

366 [-1; 0 - 30 000]

Unit of measure

Days

If the recovery history file is not needed to keep track of backups, restores, and loads, this parameter can be set to a small number.

If **rec_his_retentn** is set to -1 and **auto_del_rec_obj** is set to OFF, the number of entries indicating full database backups, and any table space backups that are associated with the database backup, corresponds with the value specified by the **num_db_backups** database configuration parameter. Other entries in the recovery history file can only be pruned by explicitly using the available commands or APIs. If **rec_his_retentn** is set to -1 and **auto_del_rec_obj** is set to ON, the history file is not automatically pruned and no recovery objects are deleted.

If **rec_his_retentn** is set to 0 and **auto_del_rec_obj** is set to OFF, all entries in the history file, except the last full backup, are pruned. If **auto_del_rec_obj** is set to ON, automated history file pruning and recovery object deletion are carried out based on the timestamp of the backup selected by the **num_db_backups** database configuration parameter.

No matter how small the retention period, the most recent full database backup plus its restore set is always kept, unless you use the **PRUNE** command with the **FORCE** option.

If both the **num_db_backups** and **rec_his_retentn** configuration parameters are set, backups are not removed unless both the **num_db_backups** and **rec_his_retentn** conditions are satisfied.

restore_pending - Restore pending

This parameter states whether a RESTORE PENDING status exists in the database.

Configuration type

Database

Parameter type

Informational

restrict_access - Database has restricted access configuration parameter

This parameter indicates whether the database was created using the restrictive set of default actions. In other words, if it was created with the **RESTRICTIVE** clause in the **CREATE DATABASE** command.

Configuration type

Database


Parameter type

Informational

YES The **RESTRICTIVE** clause was used in the **CREATE DATABASE** command when this database was created.

NO The **RESTRICTIVE** clause was not used in the **CREATE DATABASE** command when this database was created.

Related information:

 Best practices: A practical guide to restrictive databases

rollfwd_pending - Roll forward pending indicator

This parameter informs you whether or not a rollforward recovery is required, and where it is required.

Configuration type

Database

Parameter type

Informational

This parameter can indicate one of the following states:

- **DATABASE**, meaning that a rollforward recovery procedure is required for this database
- **TABLESPACE**, meaning that one or more table spaces need to be rolled forward
- **NO**, meaning that the database is usable and no rollforward recovery is required.

The recovery (using **ROLLFORWARD DATABASE**) must complete before you can access the database or table space.

section_actuals - Section actuals configuration parameter

Section actuals are runtime statistics that are measured during section execution. This parameter enables collection of section actuals such that the statistics can be viewed when an event monitor is subsequently created.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Unit of work boundary

Default [range]

NONE [NONE, BASE]

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **section_actuals** is set to the value on member 0.

If **section_actuals** is set to **NONE**, the collection of section actuals is disabled.

If **section_actuals** is set to **BASE**, section actual collection is enabled. The following statistics are collected:

- Basic operator cardinality counts

- Statistics for each object that is referenced during section execution (DML statements only)

After section actuals are enabled, capture section actuals by using an activity event monitor and view them through a section explain that you perform by using the `EXPLAIN_FROM_ACTIVITY` procedure.

self_tuning_mem- Self-tuning memory

This parameter determines whether the memory tuner will dynamically distribute available memory resources as required between memory consumers that are enabled for self-tuning.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Immediate

Default [range]

Non-partitioned environments

OFF [ON; OFF]

Partitioned environments

OFF [ON; OFF]

In a database that is upgraded from an earlier version, **self_tuning_mem** will retain the old value.

Note: The default value is subject to change by the Db2 Configuration Advisor as part of database creation.

Because memory is being traded between memory consumers, there must be at least two memory consumers enabled for self-tuning in order for the memory tuner to be active. When **self_tuning_mem** is set to ON, but there are less than two memory consumers enabled for self-tuning, the memory tuner is inactive. (The exception to this is the sort heap memory area, which can be tuned regardless of whether other memory consumers are enabled for self-tuning or not.)

This parameter is ON by default in single database partition environments. In multi-database partition environments, it is OFF by default.

The memory consumers that can be enabled for self-tuning include:

- Package cache (controlled by the **pckcachesz** configuration parameter)
- Lock List (controlled by the **locklist** and **maxlocks** configuration parameters)
- Sort heap (controlled by the **sheapthres_shr** and **sortheap** configuration parameters)
- Database shared memory (controlled by the **database_memory** configuration parameter)
- Buffer pools (controlled by the size parameter of the ALTER BUFFERPOOL and CREATE BUFFERPOOL statements)

Note: In Db2 pureScale environments, only the local buffer pools (LBPs) can be managed using the self-tuning memory feature. Memory for group buffer pools (GBPs) is allocated and controlled using the **cf_gbp_sz** configuration parameter.

To view the current setting for this parameter, use the **GET DATABASE CONFIGURATION** command specifying the **SHOW DETAIL** parameter. The possible settings returned for this parameter are:

Self Tuning Memory	(SELF_TUNING_MEM) = OFF
Self Tuning Memory	(SELF_TUNING_MEM) = ON (Active)
Self Tuning Memory	(SELF_TUNING_MEM) = ON (Inactive)
Self Tuning Memory	(SELF_TUNING_MEM) = ON

The following values indicate:

- ON (Active) - the memory tuner is actively tuning the memory on the system
- ON (Inactive) - that although the parameter is set ON, self-tuning is not occurring because there are less than two memory consumers enabled for self-tuning, or the database or instance is in quiesce mode.
- ON without (Active) or (Inactive) - from a query without the **SHOW DETAIL** option, or without a database connection.

In partitioned environments, the **self_tuning_mem** configuration parameter will only show ON (Active) for the database partition on which the tuner is running. On all other nodes **self_tuning_mem** will show ON (Inactive). As a result, to determine if the memory tuner is active in a partitioned database, you must check the **self_tuning_mem** parameter on all database partitions.

If you have upgraded to Db2 Version 9 from an earlier version of Db2 and you plan to use the self-tuning memory feature, you should configure the following health indicators to disable threshold or state checking:

- Shared Sort Memory Utilization - **db.sort_shrmem_util**
- Percentage of sorts that overflowed - **db.spilled_sorts**
- Long Term Shared Sort Memory Utilization - **db.max_sort_shrmem_util**
- Lock List Utilization - **db.locklist_util**
- Lock Escalation Rate - **db.lock_escal_rate**
- Package Cache Hit Ratio - **db.pkgcache_hitratio**

One of the objectives of the self-tuning memory feature is to avoid having memory allocated to a memory consumer when it is not immediately required. Therefore, utilization of the memory allocated to a memory consumer might approach 100% before more memory is allocated. By disabling these health indicators, you will avoid unnecessary alerts triggered by the high rate of memory utilization by a memory consumer.

Instances created in Db2 Version 9 will have these health indicators disabled by default.

seqdetect - Sequential detection and readahead flag

This parameter controls whether the database manager is allowed to perform sequential detection or readahead prefetching during I/O activity.

Configuration type

Database

Parameter type

Configurable online

Propagation class

Immediate

Default [range]

Yes [Yes; No]

The database manager can monitor I/O during index scans, and if sequential page reading is occurring the database manager can activate I/O prefetching. This type of sequential prefetch is known as *sequential detection*. However, at runtime, the prefetching type might switch from sequential detection prefetching to readahead prefetching when it detects that sequential detection prefetching is not working well enough.

If this parameter is set to No, both sequential detection and readahead prefetching are disabled for the **INSPECT** command with the **INDEXDATA** option, the **RUNSTATS** command, and all index scans during query execution. However, for the **REORG** command with the **CLEANUP** option for indexes, setting **seqdetect** to No disables only sequential detection prefetching for indexes.

Recommendation: In most cases, you should use the default value for this parameter. Try turning **seqdetect** off, only if other tuning efforts were unable to correct serious query performance problems.

sheapthres_shr - Sort heap threshold for shared sorts

This parameter represents a soft limit on the total amount of shared sort memory reservation available to sort heap-based operations.

Configuration type

Database

Parameter type

- Configurable online (requires a database connection)
- Configurable by member in a Db2 pureScale environments and in partitioned database environments.

Propagation class

Immediate

Default [range]**32-bit platforms**

AUTOMATIC [250 - 524 288]

The default setting is 5000.

64-bit platforms

AUTOMATIC [250 - 2 147 483 647]

The default setting is 5000.

Note: The default value is subject to change by the Db2 Configuration Advisor as part of database creation, which is run by default in a nonpartitioned database environment. The Db2 Configuration Advisor sets the value of the **sheapthres_shr** parameter to AUTOMATIC (minimum value of 5000) unless **DB2_WORKLOAD=ANALYTICS**, in which case it is set to a minimum value of 1000000 (not AUTOMATIC).

Unit of measure

Pages (4 KB)

AUTOMATIC

The AUTOMATIC setting is used to enable self-tuning of the **sheapthres_shr** parameter, allowing STMM to dynamically size the total shared sort memory available as workload requirements change. It is only effective under the default shared sort memory model and when **SELF_TUNING_MEM** = ON. Otherwise, the underlying configured value reflects a fixed value.

When self-tuning of the **sheapthres_shr** parameter is enabled, any attempt to manually update the underlying configured value is temporary, since the setting continues to be tuned by STMM.

While it is possible to enable self-tuning for the **sortheap** configuration parameter when the **sheapthres_shr** parameter set to a fixed value, **sortheap** cannot be set to a fixed value if the **sheapthres_shr** parameter is set to AUTOMATIC. The **sortheap** parameter is also updated to the AUTOMATIC setting when **sheapthres_shr** is updated to AUTOMATIC. Any attempt to update the **sortheap** parameter to a fixed value when the **sheapthres_shr** parameter is set to AUTOMATIC fails with an error message.

When set to AUTOMATIC, self-tuning of **sheapthres_shr** by the STMM is enabled, subject to other configuration requirements. The following conditions must be true to allow self-tuning of **sheapthres_shr** to occur:

- STMM is enabled (**SELF_TUNING_MEM**=ON).
- Shared sort memory model is enabled (**sheapthres** is set to 0).
- The **sortheap** parameter is set to AUTOMATIC.
- **DB2_WORKLOAD=analytics** is not set.
- For a partitioned database environment, the database is explicitly activated.

Self-tuning of the **sort** parameter is not supported for workloads that access column-organized tables, and is disabled when **DB2_WORKLOAD** =ANALYTICS. In such cases, set the value of the **sheapthres_shr** parameter to a fixed value, not AUTOMATIC, otherwise suboptimal performance or out-of-memory conditions might occur. When the value of the **DB2_WORKLOAD** registry variable is set to ANALYTICS, you cannot set the **sheapthres_shr** parameter to AUTOMATIC, and the Db2 Configuration Advisor automatically configures a fixed value for the **sheapthres_shr** parameter.

Various runtime operations, including those runtime operations that are not technically sort operations, allocate working memory that is based on the **sortheap** setting. The following operations allocate working memory that is based on the **sortheap** setting:

- Sort
- Hash join
- Index ANDing
- Block index ANDing
- Table in memory
- Merge join
- Scalar aggregation
- Partial early distinct and early aggregation operations
- Table queues

- Hashed GROUP BY
- column-organized data processing

While the **sortheap** parameter setting guides the maximum memory usage per operation, the **sheapthres_shr** parameter setting guides the overall memory available to sort memory consumers per database per member. The **sheapthres_shr** parameter guides the overall memory that is available by reducing the amount of sort reservation that is allowed as the total reservation requested by all activities on the database approaches the **sheapthres_shr** value. When the total reservation requested reaches the **sheapthres_shr** value, only minimum reservations are allowed, and performance might degrade. When the total reservation reaches 1.25 times the **sheapthres_shr** value, requests for sort memory might be denied and an error (SQL0955C) is returned to the application.

There are three sort memory models possible. The model in use depends upon a number of elements in the configuration.

Shared sort memory model

The shared sort memory model is the default model and is in effect whenever **sheapthres** = 0. The **sheapthres** setting guides throttling for the private sort model, and a setting of 0 disables private sort memory. Under the shared sort model, all **sortheap** allocations are from the shared sort heap (**sheapthres_shr**), which is part of database shared memory (**database_memory**). The shared sort memory model is the only model where STMM tuning of **sortheap** and **sheapthres_shr** can occur.

Private sort memory model

The private sort memory model is active whenever **sheapthres** is not equal to zero and the configuration also does not enable shared sort memory. Under the private sort memory model, **sortheap** allocations are only allocated from private memory. Operations specifically requiring shared sort memory are not valid and return errors. No STMM sort-tuning takes place under this model.

Hybrid sort memory model

The hybrid sort memory model is active whenever **sheapthres** is not equal to zero, but the configuration dictates that shared sort memory is made available for certain operations. Operations not requiring shared sort memory are allocated from private memory. No STMM sort-tuning takes place under this model.

Any one of the following configuration settings can enable shared sort memory:

- Intra-parallelism is enabled (INTRA_PARALLEL = YES)
- Connection Concentrator is enabled (MAX_CONNECTIONS > MAX_COORDAGENTS)
- DB2_WORKLOAD=ANALYTICS

If shared sort memory is not enabled, the shared sort memory model and the hybrid sort memory models are not active and the following operations fail:

- Loading data into an XML table
- Column-organized query processing
- Applications requesting intra-parallel processing

If any of the operations in the preceding list fails, enable the shared sort memory model by completing all of the following steps:

- Set the **sheapthres** configuration parameter to 0.
- Recycle the Db2 instance. To recycle the Db2 instance, run the **db2stop** command followed by the **db2start** command.
- Configure the **sheapthres_shr** parameter appropriately.

Monitoring

There are numerous monitoring elements available.

For general monitoring, you can use the following monitoring elements:

- ACTIVE_SORTS
- TOTAL_SECTION_SORT_TIME
- TOTAL_SECTION_SORT_PROC_TIME
- TOTAL_SECTION_SORTS
- TOTAL_SORTS

For monitoring of constrained total sort memory configuration, you can use the following monitoring elements:

- POST_THRESHOLD_SORTS (for shared sorts)
- POST_THRESHOLD_PEDS
- POST_THRESHOLD_PEAS
- POST_SHRTHRESHOLD_HASH_JOINS (for shared sorts)
- POST_THRESHOLD_HASH_GRPBY
- POST_THRESHOLD_OLAP_FUNCS
- SORT_OVERFLOW
- TQ_SORT_HEAP_REJECTIONS
- SORT_HEAP_ALLOCATED (shared sort reservations)
- SORT_SHRHEAP_TOP (high water mark for shared sort reservations)

The SORT_HEAP_ALLOCATED and SORT_SHRHEAP_ALLOCATED monitoring elements reflect reservation requests, not the actual amount of memory that is allocated. It is normal for operations to not fully allocate all of the requested reservation amounts.

For monitoring of shared sort reservation levels, use the MON_GET_DATABASE routine. The following example shows a query that can be used to monitor the shared sort reservation levels:

```
select SORT_SHRHEAP_ALLOCATED, SORT_SHRHEAP_TOP from table (MON_GET_DATABASE (null))
```

This query returns the memory reservation levels in 4K units:

```
SORT_SHRHEAP_ALLOCATED SORT_SHRHEAP_TOP
-----
                128411                396405
```

For monitoring of shared sort memory usage, use the MON_GET_MEMORY_POOL routine. The following example shows a query that can be used to monitor the shared sort memory usage:

```
select memory_pool_used, memory_pool_used_hwm
from table (mon_get_memory_pool(null,null,null))
where memory_pool_type='SHARED_SORT'
```

This query returns the memory allocation levels for shared sort in 1K units:

MEMORY_POOL_USED	MEMORY_POOL_USED_HWM
----- 140574	----- 140574

There is only one shared sort memory pool for all applications that run on a single database.

smtp_server - SMTP server

This parameter identifies a simple mail transfer protocol (SMTP) server. This SMTP server transmits email sent by the UTL_MAIL built-in module.

The parameter also accepts a comma-delimited list of SMTP servers. UTL_MAIL attempts to send email through the first SMTP server in the list. If that SMTP server is unavailable, the next server in the list is used. If all servers in the comma-delimited list are unreachable, an error is returned.

Configuration type

Database

Applies to

Database server with local and remote clients

Database server with local clients

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Null [comma-delimited list of valid SMTP server TCP/IP hostnames]

softmax - Recovery range and soft checkpoint interval

This parameter determines the frequency of soft checkpoints and the recovery range, which help out in the crash recovery process.

Important: The **softmax** database configuration parameter is deprecated is deprecated in Version 10.5 and might be removed in a future release. For more information, see *Some database configuration parameters are deprecated in What's New for Db2 Version 10.5*.

The **softmax** parameter is replaced with the new **page_age_trgt_mcr** and **page_age_trgt_gcr** parameters, which are both configured as a number of seconds.

Existing upgraded databases continue to use the **softmax** parameter. You can check if you are using **softmax** by querying the database configuration and checking the value of this parameter. You can switch from using **softmax** to these new parameters by setting the value of **softmax** to 0.

New databases are created with the value of **softmax** set to 0 by default.

Configuration Type

Database

Parameter Type

Configurable

Default [range]

Db2 pureScale environment

0 [1 - 65 535]

Outside of a Db2 pureScale environment

0 [1 - 100 * **logprimary**]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of Measure

Percentage of the size of one primary log file

This parameter is used to:

- Influence the number of log files that need to be recovered following a crash (such as a power failure). For example, if the default value of 100 is used, the database manager will try to keep the number of log files that need to be recovered to 1. If you specify 300 as the value of this parameter, the database manager will try to keep the number of log files that need to be recovered to 3. To influence the number of log files required for crash recovery, the database manager uses this parameter to trigger the page cleaners to ensure that pages older than the specified recovery window are already written to disk.
- Determine the frequency of soft checkpoints. It is the process of writing information to the log control file. This information is used to determine the starting point in the log in case a database restart is required.

At the time of a database failure resulting from an event such as a power failure, there might have been changes to the database which:

- Have not been committed, but updated the data in the buffer pool
- Have been committed, but have not been written from the buffer pool to the disk
- Have been committed and written from the buffer pool to the disk.

When a database is restarted, the log files will be used to perform a crash recovery of the database which ensures that the database is left in a consistent state (that is, all committed transactions are applied to the database and all uncommitted transactions are not applied to the database).

To determine which records from the log file need to be applied to the database, the database manager uses information recorded in a log control file. (The database manager actually maintains two copies of the log control file, `SQLLOGCTL.LFH.1` and `SQLLOGCTL.LFH.2`, so that if one copy is damaged, the database manager can still use the other copy.) These log control files are periodically written to disk, and, depending on the frequency of this event, the database manager might be applying log records of committed transactions or applying log records that describe changes that have already been written from the buffer pool to disk. These log records have no impact on the database, but applying them introduces some additional processing time into the database restart process.

The log control files are always written to disk when a log file is full, and during soft checkpoints. You can use this configuration parameter to control the frequency of soft checkpoints.

The timing of soft checkpoints is based on the difference between the “current state” and the “recorded state”, given as a percentage of the `logfilsiz`. The “recorded state” is determined by the oldest valid log record indicated in the log control files on disk, while the “current state” is determined by the log control

information in memory. (The oldest valid log record is the first log record that the recovery process would read.) The soft checkpoint will be taken if the value calculated by the following formula is greater than or equal to the value of this parameter:

$$(\text{space between recorded and current states}) / \text{logfilesiz}) * 100$$

Recommendation: You might want to increase or reduce the value of this parameter, depending on whether your acceptable recovery window is greater than or less than one log file. Lowering the value of this parameter will cause the database manager both to trigger the page cleaners more often and to take more frequent soft checkpoints. These actions can reduce both the number of log records that need to be processed and the number of redundant log records that are processed during crash recovery.

Note however, that more page cleaner triggers and more frequent soft checkpoints increase the processing time associated with database logging, which can impact the performance of the database manager. Also, more frequent soft checkpoints might not reduce the time required to restart a database, if you have:

- Very long transactions with few commit points.
- A very large buffer pool and the pages containing the committed transactions are not written back to disk very frequently. (Note that the use of asynchronous page cleaners can help avoid this situation.)

In both of these cases, the log control information kept in memory does not change frequently and there is no advantage in writing the log control information to disk, unless it has changed.

sortheap - Sort heap size

This parameter defines the maximum number of private or shared memory pages that an operation that requires sort heap memory allocates.

Configuration type

Database

Parameter type

- Configurable online (requires a database connection)
- Configurable by member in a Db2 pureScale environment and in partitioned database environments.

Propagation class

Immediate

Default [range]

32-bit platforms

AUTOMATIC [16 - 524 288]

The default value is 256.

64-bit platforms

AUTOMATIC [16 - 4 294 967 295]

The default value is 256.

Note: The default value is subject to change by the Db2 Configuration Advisor as part of database creation, which is run by default in a nonpartitioned database environment. The Db2 Configuration Advisor sets

the value of the **sortheap** parameter to AUTOMATIC (minimum value of 256) unless DB2_WORKLOAD=ANALYTICS, in which case it is set to a minimum value of 32768 (not AUTOMATIC).

Unit of measure

Pages (4 KB)

When allocated

As needed to run operations that require sort memory.

When freed

Some memory might be freed dynamically when no longer needed. All remaining memory is freed when the operation that requires sort memory completes.

AUTOMATIC

The AUTOMATIC setting is used to enable self-tuning of **sortheap** by STMM, and is only effective under the default shared sort memory model. Otherwise, the underlying configured value reflects a fixed value.

The AUTOMATIC setting enables the simulation processing that is required to support STMM tuning, and when STMM is enabled self-tuning of the **sortheap** setting. When self-tuning of **sortheap** is enabled, any attempt to manually update the numeric value while you maintain the automatic setting is temporary since the setting continues to be tuned by STMM.

While it is possible to enable self-tuning for the **sortheap** parameter and keep the **sheapthres_shr** parameter set to a fixed value, **sortheap** cannot be set to a fixed value if **sheapthres_shr** is set to AUTOMATIC. The **sortheap** value is also updated to the AUTOMATIC setting when the **sheapthres_shr** setting is updated to AUTOMATIC, and any attempt to update **sortheap** to a fixed value when **sheapthres_shr** is AUTOMATIC fails with SQL5147N.

When set to AUTOMATIC, self-tuning of **sheapthres_shr** by the STMM is enabled, subject to other configuration requirements. The following conditions must be true to allow self-tuning of **sheapthres_shr** to occur:

- STMM is enabled (SELF_TUNING_MEM=ON).
- Shared sort memory model is enabled (**sheapthres** is set to 0).
- The **sortheap** parameter is set to AUTOMATIC.
- DB2_WORKLOAD=ANALYTICS is not set.
- For a partitioned database environment, the database is explicitly activated.

Self-tuning of the **sort** parameter is not supported for workloads that access column-organized tables, and is disabled when **DB2_WORKLOAD**=ANALYTICS. In such cases, set the value of the **sortheap** parameter to a fixed value, not AUTOMATIC, otherwise suboptimal performance or out-of-memory conditions might occur. When the value of the **DB2_WORKLOAD** registry variable is set to ANALYTICS, you cannot set the **sortheap** parameter to AUTOMATIC, and the Db2 Configuration Advisor automatically configures a fixed value for the **sortheap** parameter.

Various runtime operations, including those operations that are not true sort operations, allocate memory that is based on the **sortheap** setting. The following operations allocate working memory that is based on the **sortheap** setting:

- Sort
- Hash join

- Index ANDing
- Block index ANDing
- Table in memory
- Merge join
- Scalar aggregation
- Partial early distinct and early aggregation operations
- Table queues
- Hashed GROUP BY
- column-organized data processing

The **sortheap** setting guides the maximum memory usage per operation. The allocated or target memory might be less than **sortheap** setting due to any of the following factors:

- The operation that is run.
- The optimizer's plan choice.
- Whether total sort memory for a member or partition is being throttled (see **sheapthres**, **sheapthres_shr**).

Multiple operations can be active concurrently during the execution of a single SQL statement, each with their own **sortheap** based allowance. The concurrency of **sortheap** consuming operations significantly increases when the intra-parallelism or the data partitioning feature is enabled, as multiple agents are typically running operations that require separate **sortheap** based memory areas.

There are three sort memory models possible. The model in use depends upon a number of elements in the configuration.

Shared sort memory model

The shared sort memory model is the default model and is in effect whenever **sheapthres** = 0. The **sheapthres** setting guides throttling for the private sort model, and a setting of 0 disables private sort memory. Under the shared sort model, all **sortheap** allocations are from the shared sort heap (**sheapthres_shr**), which is part of database shared memory (**database_memory**). The shared sort memory model is the only model where STMM tuning of **sortheap** and **sheapthres_shr** can occur.

Private sort memory model

The private sort memory model is active whenever **sheapthres** is not equal to zero and the configuration also does not enable shared sort memory. Under the private sort memory model, **sortheap** allocations are only allocated from private memory. Operations specifically requiring shared sort memory are not valid and return errors. No STMM sort-tuning takes place under this model.

Hybrid sort memory model

The hybrid sort memory model is active whenever **sheapthres** is not equal to zero, but the configuration dictates that shared sort memory is made available for certain operations. Operations not requiring shared sort memory are allocated from private memory. No STMM sort-tuning takes place under this model.

Any one of the following configuration settings can enable shared sort memory:

- Intra-parallelism is enabled (INTRA_PARALLEL = YES)

- Connection Concentrator is enabled (MAX_CONNECTIONS > MAX_COORDAGENTS)
- DB2_WORKLOAD=ANALYTICS

If shared sort memory is not enabled, the shared sort memory model and the hybrid sort memory models are not active and the following operations fail:

- Loading data into an XML table
- Column-organized query processing
- Applications requesting intra-parallel processing

If any of the operations in the preceding list fail, enable the shared sort memory model by completing all of the following steps:

- Set the **sheapthres** configuration parameter to 0.
- Recycle the Db2 instance. To recycle the Db2 instance, run the **db2stop** command followed by the **db2start** command.
- Configure the **sheapthres_shr** parameter appropriately.

Recommendation

When you are working with the **sortheap** setting, you must consider the following factors:

- Appropriate indexes can minimize the use of the sort heap.
- Increase the size of the **sortheap** parameter when frequent large sorts are required.
- When not using STMM to tune the **sortheap** parameter, update the **sortheap** setting to a numeric value, either by updating to a fixed value or MANUAL. Updating the **sortheap** parameter to a numeric value avoids the small performance cost of automatic sort simulation
- When you increase the value of the **sortheap** parameter, you must examine whether the **sheapthres** and **sheapthres_shr** parameters in the database manager configuration file also need to be adjusted.
- Ensure the ratio of **sortheap:sheapthres_shr** or **sortheap:sheapthres** is sufficient to avoid triggering throttling of **sortheap** allowances, especially in parallelized environments that use intra-parallelism or the data partitioning feature.
- The **sortheap** size is used by the optimizer in determining access paths. You must consider rebinding applications by using the **REBIND** command after you change the size of the **sortheap** parameter.

When the **sortheap** value is updated dynamically, the database manager immediately starts to use this new value for any current or new operations.

Monitoring

There are numerous monitoring elements available.

For general monitoring, you can use the following monitoring elements:

- ACTIVE_SORTS
- TOTAL_SECTION_SORT_TIME
- TOTAL_SECTION_SORT_PROC_TIME
- TOTAL_SECTION_SORTS
- TOTAL_SORTS

For monitoring of constrained total sort memory configuration, you can use the following monitoring elements:

- POST_THRESHOLD_SORTS (for private sorts only)
- POST_SHRTHRESHOLD_SORTS (for shared sorts only)
- POST_THRESHOLD_PEDS
- POST_THRESHOLD_PEAS
- POST_SHRTHRESHOLD_HASH_JOINS (for shared hash joins only)
- POST_THRESHOLD_HASH_JOINS (for private hash joins only)
- POST_THRESHOLD_HASH_GRPBY
- POST_THRESHOLD_OLAP_FUNCS
- SORT_OVERFLOW
- TQ_SORT_HEAP_REJECTIONS
- SORT_HEAP_ALLOCATED (private sort reservations)
- SORT_SHRHEAP_TOP (high water mark for shared sort reservations)

The SORT_HEAP_ALLOCATED, SORT_SHRHEAP_ALLOCATED, SORT_SHRHEAP_TOP monitoring elements reflect reservation requests by sort heap-based operators, not the actual amount of memory that is allocated. It is normal for operations to not fully allocate all of the requested reservation amounts.

For monitoring of sort memory usage, use the MON_GET_MEMORY_POOL routine.

For more information about monitoring sort memory usage, see “sheapthres - Sort heap threshold” on page 92 and “sheapthres_shr - Sort heap threshold for shared sorts” on page 233.

sql_ccflags - Conditional compilation flags

This parameter contains a list of conditional compilation values for use in conditional compilation of selected SQL statements.

Configuration type

Database

Parameter type

Configurable Online

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **sql_ccflags** is set to the value on member 0.

The value of **sql_ccflags** must include one or more name and value pairs, where the name is separated from the value by the colon character. Each name and value pair must be separated from the previous pair by a comma. The name must be a valid ordinary SQL identifier. The value must be an SQL BOOLEAN constant, an SQL INTEGER constant, or the NULL keyword. The maximum length of the string is 1023 bytes.

When the value of **sql_ccflags** is updated, the value is not immediately checked to ensure that it is valid. Checking occurs the first time that the value is used to initialize the CURRENT SQL_CCFLAGS special register; that is, when a connection to the database first references CURRENT SQL_CCFLAGS, or when an inquiry directive is encountered in an SQL statement. After updating the value of

sql_ccflags, connect to the database and query the special register by using the following statement: `VALUES CURRENT SQL_CCFLAGS`.

stat_heap_sz - Statistics heap size

This parameter indicates the *maximum* size of the heap used in collecting statistics using the **RUNSTATS** command.

The constraint set by this parameter applies to each **RUNSTATS** operation.

With Version 9.5, this database configuration parameter has a default value of **AUTOMATIC**, meaning that it increases as needed until either the **app1_memory** limit is reached, or the **instance_memory** limit is reached.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]

32-bit platforms

Automatic [1 096 - 524 288]

64-bit platforms

Automatic [1 096 - 2 147 483 647]

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

When the **RUNSTATS** utility is started

When freed

When the **RUNSTATS** utility is completed

Recommendation: **RUNSTATS** memory requirements depend on several factors. More memory is required with more statistic options, for example if **LIKE** statistics or **DETAILED** index statistics are being collected. When column statistics are collected, gathering statistics of a higher number of columns will require more memory. When distribution statistics are being collected, gathering a higher number of frequent, quantile values, or both will require more memory. The default setting of **AUTOMATIC** is recommended.

stmt_conc - Statement concentrator configuration parameter

This configuration parameter sets the default statement concentrator behavior.

Configuration type

Database

Parameter type

- Configurable online

Propagation class

Statement boundary

Default [range]
OFF [OFF, LITERALS]

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **stmt_conc** is set to the value on member 0.

This configuration parameter enables statement concentration for dynamic statements. The setting in the database configuration is used only when the client does not explicitly enable or disable the statement concentrator.

When enabled, the statement concentrator modifies dynamic statements to allow increased sharing of package cache entries.

The statement concentrator is disabled when the configuration parameter is set to OFF. When the configuration parameter is set to LITERALS, the statement concentrator is enabled. When the statement concentrator is enabled, SQL statements that are identical, except for the values of literals in the statements, might share package cache entries.

For example, when **stmt_conc** is set to LITERALS, the following statements share an entry in the package cache

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO='000020'  
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO='000070'
```

The entry in the package cache uses the following statement:

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO=:L0
```

The Db2 database system provides the value for :L0 based on the literal used in the original statements:

```
:L0(either '000020' or '000070')
```

This parameter can have a significant impact on access plan selection because it alters the statement text. The statement concentrator must be used only when similar statements in the package cache have similar plans. For example, if different literal values in a statement result in different plans, then statement concentrator must not be set to LITERALS.

The **stmt_conc** configuration parameter might cause the length attributes for VARCHAR and VARGRAPHIC string literals to be greater than the length of the string literal.

The statement concentrator might cause some built-in functions to return different result types. For example, REPLACE can return a different type when statement concentrator is used. The WORKDEPT column is defined as CHAR(3), the following query returns VARCHAR(3) when statement concentrator is disabled:

```
SELECT REPLACE(WORKDEPT,'E2','E9') FROM EMPLOYEE
```

When **stmt_conc**=LITERALS, the two string literals are replaced with parameter markers and the return type is VARCHAR(6).

stmtheap - Statement heap size

This parameter specifies the limit of the statement heap, which is used as a work space for the SQL or XQuery compiler during compilation of an SQL or XQuery statement.

Configuration type

Database

Parameter type

Configurable Online

Configurable by member in a Db2 pureScale environment

Propagation class

Statement boundary

Default [range]

For 32-bit platforms

AUTOMATIC [128 - 524288]

- Database server with local and remote clients: the default value is AUTOMATIC with an underlying value of 2048.
- This parameter can also be set to a fixed value only, without the AUTOMATIC attribute.

For 64-bit platforms

AUTOMATIC [128 - 2 147 483 647]

- Database server with local and remote clients: the default value is AUTOMATIC with an underlying value of 8192.
- This parameter can be set to a fixed value only, without the AUTOMATIC attribute.

Note: The default value is subject to change by the Db2 Configuration Advisor after initial database creation.

Unit of measure

Pages (4 KB)

When allocated

As required for each statement during precompiling or binding

When freed

When precompiling or binding of a statement is complete

The statement heap does not stay permanently allocated. It is allocated and released for every SQL or XQuery statement handled. For dynamic SQL or XQuery statements, the statement heap is used during the execution of your program. For static SQL or XQuery statements, the statement heap is used during the bind process, but not during program execution.

The **stmtheap** parameter can be set to AUTOMATIC with an underlying value or a fixed value.

When the **stmtheap** parameter is set to AUTOMATIC, the underlying value enforces a limit on the amount of memory allocated for a single compilation using dynamic join enumeration. If a memory limit is encountered, the statement compilation restarts using greedy join enumeration and an unlimited statement heap. With the AUTOMATIC option, while the greedy join enumeration is performed, statement compilation is only limited by the amount of remaining application memory

(**APPL_MEMORY**), instance memory (**INSTANCE_MEMORY**), or system memory. If greedy join enumeration is successfully completed, an SQL0437W warning is returned to the application. If a greedy join enumeration encounters a memory limit, the statement preparation fails with SQL0101N.

For example, **db2 update db cfg for SAMPLE using STMHEAP 8192 AUTOMATIC** creates a statement heap limit of 8192 * 4K (32MB) for dynamic join enumeration and no limit for greedy join enumeration.

When the **stmheap** parameter is set to a fixed value, the limit applies to both dynamic and greedy join enumeration. If dynamic join enumeration encounters a memory limit, a greedy join enumeration is attempted with the same fixed statement heap limit. If greedy join enumeration is successfully completed, an SQL0437W warning is returned to the application. If a greedy join enumeration encounters a memory limit, the statement preparation fails with SQL0101N.

For example, **db2 update db cfg for SAMPLE using STMHEAP 8192** creates a statement heap limit of 8192 * 4K (32MB) for both dynamic and greedy join enumeration.

If the runtime performance of your query is not sufficient, consider increasing the **stmheap** configuration parameter value (either the value underlying **AUTOMATIC** or a fixed value) to ensure that dynamic programming join enumeration is successful. If you update the **stmheap** configuration parameter to improve the performance of a query, force the statement to recompile. Forcing a statement to recompile might cause the query optimizer to create a new access plan that takes advantage of the new statement heap size.

Note: Dynamic programming join enumeration occurs only at optimization classes 3 and higher. The default optimization class is 5.

string_units - Default string units

This parameter specifies the default string units that are used when defining character data types and graphic data types in Unicode databases.

Configuration type

Database

Parameter type

Configurable

Default [range]

SYSTEM [SYSTEM, CODEUNITS32]

The value of this parameter is used only when the session level global variable *NLS_STRING_UNITS* is null. The default value of *NLS_STRING_UNITS* is null if it is not set during that session.

When the **string_units** configuration parameter is set to **SYSTEM** in a Unicode database:

- CHAR, VARCHAR, and CLOB data types that are defined without specifying the **CODEUNITS32** keyword default to **OCTETS**.
- GRAPHIC, VARGRAPHIC, and DBCLOB data types that are defined without specifying the **CODEUNITS32** keyword default to **CODEUNITS16**.

When the **string_units** configuration parameter is set to SYSTEM in a non-Unicode database:

- CHAR, VARCHAR, and CLOB data types that are defined without specifying the CODEUNITS32 keyword default to OCTETS.
- GRAPHIC, VARGRAPHIC, and DBCLOB data types that are defined without specifying the CODEUNITS32 keyword have implicit string units of double bytes.

The **string_units** configuration parameter can be set to CODEUNITS32 in Unicode databases only. When the **string_units** configuration parameter is set to CODEUNITS32:

- CHAR, VARCHAR, and CLOB data types that are defined without specifying the BYTE or OCTETS keywords default to CODEUNITS32.
- GRAPHIC, VARGRAPHIC, and DBCLOB data types that are defined without specifying the CODEUNITS16 keyword default to CODEUNITS32.

suspend_io - Database I/O operations state configuration parameter

This parameter shows whether the I/O write operations for a database are suspended or are being suspended.

Configuration type

Database

Parameter type

Informational

Values are YES, IN_PROGRESS, and NO.

system_time_period_adj - Adjust temporal SYSTEM_TIME period database configuration parameter

This database configuration parameter specifies what action to take when a history row for a system-period temporal table is generated with an end timestamp that is less than the begin timestamp.

This can happen when two different transactions conflict in their attempt to update the same row in a system-period temporal table. It can also happen as the result of an adjustment to the system clock; for example, if the system clock is adjusted back an hour for the end of daylight savings time.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable online

Propagation class

Immediate

Default [range]
NO [NO, YES]

When a row is updated in a system-period temporal table, a history row is generated with a `SYSTEM_TIME` period indicating the range of time when the data in the history row was current. The value in the row-begin column indicates when the data in the history row became current. The value in the row-end column indicates when the history row became history data.

Following is an example of how two conflicting transactions could potentially generate a history row that has an row-end timestamp that is less than the row-begin timestamp.

1. Transaction TRA has a row-begin value generated for a row in a system-period temporal table it is updating at timestamp T1.
2. Transaction TRB has a row-begin value generated for the same row that it is updating at timestamp T2 (where $T1 < T2$).
3. Transaction TRB generates a history row and commits.
4. Transaction TRA generates its history row and commits.

After this sequence of events, the history row generated for transaction TRA would have an end timestamp that is less than its begin timestamp.

The database manager can ensure that generated history rows always have an end timestamp greater than the start timestamp by allowing timestamp adjustments when there are conflicts or by rolling back one of the transactions involved.

NO No timestamp value adjustments are made when the end timestamp is less than the start timestamp for a history row that is being inserted. Instead, the transaction that is attempting to insert the history row fails and an error is returned (SQLSTATE 57062, SQLCODE SQL20528N). Not allowing adjustments ensures that all history rows generated during the transaction have the same end timestamp and can easily be identified using that end timestamp.

YES An adjustment is made to the timestamp value of the row-begin column value for the system-period temporal table and the end timestamp value for the generated history row when there are timestamp conflicts. The adjustment consists of modifying the end timestamp to be greater than the start timestamp by 1 microsecond. This ensures that the end timestamp is greater than the start timestamp for the history row. A message is returned indicating that an adjustment was made (SQLSTATE 01695, SQLCODE SQL5191W).

When no timestamp adjustments are necessary, SQLCODE DB20000I is returned.

Application programmers might consider using SQLCODE or SQLSTATE values to handle these timestamp value adjustment-related return codes from SQL statements.

territory - Database territory

This parameter shows the territory used to create the database. **territory** is used by the database manager when processing data that is territory sensitive.

Configuration type
Database

Parameter type
Informational

trackmod - Track modified pages enable

This parameter specifies whether the database manager will track database modifications so that the backup utility can detect which subsets of the database pages must be examined by an incremental backup and potentially included in the backup image.

Configuration type
Database

Parameter type
Configurable

Default [range]
No [Yes, No]

After setting this parameter to "Yes", you must take a full database backup in order to have a baseline against which incremental backups can be taken. Also, if this parameter is enabled and if a table space is created, then a backup must be taken which contains that table space. This backup could be either a database backup or a table space backup. Following the backup, incremental backups will be permitted to contain this table space.

tsm_mgmtclass - Tivoli Storage Manager management class

The Tivoli Storage Manager management class determines how the TSM server manages the backup versions of the objects being backed up.

Configuration type
Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]
NULL [YOUR_STRING_VALUE]

By default, no management class is specified by a Db2 database.

When performing any TSM backup, before using the management class specified by the database configuration parameter, TSM first attempts to bind the backup object to the management class specified in the INCLUDE-EXCLUDE list found in the TSM client options file. If a match is not found, the default TSM management class specified on the TSM server is used. TSM then rebinds the backup object to the management class specified by the database configuration parameter.

The default management class, as well as the management class specified by the database configuration parameter, must contain a backup copy group, or the backup operation might fail.

tsm_nodename - Tivoli Storage Manager node name

This parameter is used to override the default setting for the node name associated with the Tivoli Storage Manager (TSM) product.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Statement boundary

Default [range]

Null [any string]

The node name is needed to allow you to restore a database that was backed up to TSM from another node.

The default is that you can only restore a database from TSM on the same node from which you did the backup. It is possible for the **tsm_nodename** to be overridden during a backup done through Db2 (for example, with the **BACKUP DATABASE** command).

tsm_owner - Tivoli Storage Manager owner name

This parameter is used to override the default setting for the owner associated with the Tivoli Storage Manager (TSM) product.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class

Statement boundary

Default [range]

Null [any string]

The owner name is needed to allow you to restore a database that was backed up to TSM from another node. It is possible for the **tsm_owner** to be overridden during a backup done through Db2 (for example, with the **BACKUP DATABASE** command).

Note: The owner name is case sensitive.

The default is that you can only restore a database from TSM on the same node from which you did the backup.

tsm_password - Tivoli Storage Manager password

This parameter is used to override the default setting for the password associated with the Tivoli Storage Manager (TSM) product.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Propagation class
Statement boundary

Default [range]
Null [any string]

The password is needed to allow you to restore a database that was backed up to TSM from another node.

Note: If the **tsm_nodename** is overridden during a backup done with Db2 (for example, with the **BACKUP DATABASE** command), the **tsm_password** might also have to be set.

The default is that you can only restore a database from TSM on the same node from which you did the backup. It is possible for the **tsm_nodename** to be overridden during a backup done with Db2.

user_exit_status - User exit status indicator

If set to YES, the **user_exit_status** parameter indicates that the database manager is enabled for rollforward recovery and that the database archives and retrieves log files based on the values set by either the **logarchmeth1** parameter or the **logarchmeth2** parameter.

Configuration type
Database

Parameter type
Informational

util_heap_sz - Utility heap size

This parameter guides the amount of memory that is allocated by the database utilities.

Configuration type
Database

Parameter type

- Configurable online (requires a database connection)
- Configurable by a member in a Db2 pureScale environment or partitioned database environment

Propagation class
Immediate

Default [range]
AUTOMATIC [16 - 2147483647]

The default value is AUTOMATIC with an underlying value of 5000. The default value is subject to change by the Db2 Configuration Advisor as part of database creation, which is run by default in a nonpartitioned database environment. The Db2 Configuration Advisor sets the **UTIL_HEAP_SZ** parameter to AUTOMATIC (minimum value of 5000) unless **DB2_WORKLOAD=ANALYTICS**, in which case it is set to AUTOMATIC (minimum value of 1000000).

Unit of measure
Pages (4 KB)

When allocated

As required by the database manager utilities

When freed

When the utility no longer needs the memory

Utility heap usage is allowed to expand as needed beyond the configured value subject to **database_memory**, **instance_memory**, and system memory limits.

Utilities can calculate their memory usage target based on the needs of the utility and the amount of available utility heap. This target can be overridden for some utilities by specifying the amount of memory to use. For example, you can specify the amount of memory to use by providing a value for the **DATA BUFFER** option of the **LOAD** utility.

By default, the **util_heap_sz** parameter is set to **AUTOMATIC**. This **AUTOMATIC** setting allows utilities to consider database memory overflow as part of the available utility heap in addition to the underlying configured value. The underlying configured value represents a reservation of database memory. When the reserved amount is fully used, more utility heap memory is taken from database memory overflow.

When the **util_heap_sz** parameter is set to a fixed value, utilities do not consider database memory overflow as part of the available utility heap. The fixed value represents a reservation of database memory and normally guides the maximum amount of memory that is used by utilities, but does not represent a hard limit. When the reserved amount is fully used, more utility heap memory is taken from the database memory overflow. Typically the reserved amount is never fully used unless the default behavior of utilities is overridden. For example, the default behavior of utilities is overridden when **BUFFER** sizes are provided for the **LOAD** or **BACKUP** commands.

Recommendation: It is recommended to use the default setting of **AUTOMATIC**. The availability of database memory overflow under the **AUTOMATIC** setting satisfies the temporary needs of utilities while avoiding the need for excessive memory reservations when utilities are not running. In addition, when utilities' memory needs increase, the self-tuning memory manager (STMM) can make extra database memory overflow available to the utility heap by tuning the overall database memory configuration.

Monitoring

You can monitor your utility heap usage by using the **MON_GET_MEMORY_POOL** table function. For example, the following query returns the values of the **memory_pool_used** and **memory_pool_used_hwm** monitor elements:

```
select memory_pool_used, memory_pool_used_hwm
from table (mon_get_memory_pool(null,null,null))
where memory_pool_type='UTILITY'
```

The following is a sample output that is generated by running the query:

```
MEMORY_POOL_USED      MEMORY_POOL_USED_HWM
-----
                137280                137280
```

1 record(s) selected.

You can also use the **db2pd** command with the **-db** *database_name* and **-mempools** parameters to monitor utility heap usage.

varchar2_compat - varchar2 compatibility database configuration parameter

This parameter indicates whether the compatibility semantics associated with the VARCHAR2 and NVARCHAR2 data types are applied to the connected database.

Configuration type

Database

Parameter type

Informational

The value is determined at database creation time, and is based on the setting of the **DB2_COMPATIBILITY_VECTOR** registry variable for VARCHAR2 support. The value cannot be changed.

vendoropt - Vendor options

This parameter specifies additional parameters that Db2 might need to use to communicate with storage systems during backup, restore, or load copy operations.

Configuration type

Database

Applies to

- Database server with local and remote clients
- Client
- Database server with local clients
- Partitioned database server with local and remote clients

Parameter type

Configurable Online

Default [range]

Null []

Restrictions

You cannot use the **vendoropt** configuration parameter to specify vendor-specific options for snapshot backup or restore operations. You must use the **OPTIONS** parameter of the backup or restore utilities instead.

In TSM environments configured to support proxy nodes, the "-fromnode=nodename" option and the "-fromowner=ownername" option are not compatible with the "-asnodename=nodename" option and cannot be used together. Use the -asnodename option for TSM configurations using proxy nodes and the other two options for other types of TSM configurations. For more information, see "Configuring a Tivoli Storage Manager client".

wlm_admission_ctrl - WLM admission control

This parameter is reserved for future use and cannot be modified to anything other than the default value.

Configuration type

Database

Parameter type

- Configurable online

Default [range]

No

wlm_agent_load_trgt - WLM agent load target

This parameter is reserved for future use and cannot be modified to anything other than the default value.

When set to AUTOMATIC, an appropriate value will be automatically set, based on the number of physical cores.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]

AUTOMATIC

wlm_collect_int - Workload management collection interval configuration parameter

This parameter specifies a collect and reset interval, in minutes, for workload management (WLM) statistics.

Configuration type

Database

Parameter type

- Configurable online

Default [range]

0 [0 (no collection performed), 5 - 32 767]

Upgrade Note

- If you are upgrading from a Db2 Version 9.8 Fix Pack 4 pureScale environment or earlier, the value of **wlm_collect_int** is set to the value on member 0.

Every x minutes, (where x is the value of the **wlm_collect_int** parameter) all workload management statistics are collected and sent to any active statistics event monitor; then the statistics are reset. If an active statistics event monitor exists, depending on how it was created, the statistics are written to a file, to a pipe, or to a table. If it does not exist, the statistics are only reset and not collected.

Collections occur at the specified interval times as measured relative to Sunday at 00:00:00. When the catalog member becomes active, the next collection occurs at the start of the next scheduled interval relative to this fixed time. The scheduled interval is not relative to the catalog member activation time. If a member is not active at the time of collection, no statistics are gathered for that member. For example, if the interval value was set to 60 and the catalog member was activated on 9:24 AM on Sunday, then the collections would be scheduled to occur each hour on the hour. Therefore, the next collection will occur at 10:00 AM. If the member is not active at 10:00 AM, then no statistics are gathered for that member.

The collect and reset process is initiated from the catalog member. The **wlm_collect_int** parameter must be specified on the catalog member. It is not used on other members.

The workload management statistics collected by a statistics event monitor can be used to monitor both short term and long term system behavior. A small interval can be used to obtain both short term and long term system behavior because the results can be merged together to obtain long term behavior. However, having to manually merge the results from different intervals complicates the analysis. If it is not required, a small interval unnecessarily increases the processing time. Therefore, reduce the interval to capture shorter term behavior, and increase the interval to reduce processing time when only analysis of long term behavior is sufficient.

The interval needs to be customized per database, not for each SQL request, or command invocation, or application. There are no other configuration parameters that need to be considered.

Note: All WLM statistics table functions return statistics that have been accumulated since the last time the statistics were reset. The statistics will be reset regularly on the interval specified by this configuration parameter.

wlm_cpu_limit - WLM CPU limit

This parameter specifies the fixed amount of CPU that can be consumed by work that is running on a database.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]

0 [0 - 100]

wlm_cpu_shares - WLM CPU shares

This parameter specifies the number of shares of CPU resources allocated for work in a database.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]

1000 [1 - 65535]

wlm_cpu_share_mode - WLM CPU share mode

Db2 workload management can manage CPU resources by using shares-based entitlements that are assigned to a database. The number of CPU shares that are assigned can be specified by using the WLM_CPU_SHARES configuration parameter. This parameter specifies the type of share.

Configuration type

Database

Parameter type

- Configurable online
- Configurable by member in a Db2 pureScale environment

Default [range]

Hard [Hard, Soft]

Usage notes

Hard Hard shares prevent work that is running on the database from using more than its share of CPU resources.

Soft Soft shares give the database the ability to use more than its share of CPU resources, if those CPU resources are unused.

Db2 Administration Server (DAS) configuration parameters

You can set Db2 Administration Server (DAS) configuration parameters to set authentication and global variables on your DAS. This type of parameter is available if you have DAS installed on your Db2 server.

authentication - Authentication type DAS

This parameter determines how and where authentication of a user takes place.

Important: The Db2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in Db2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “Db2 administration server (DAS) has been deprecated” at .

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable

Default [range]

SERVER_ENCRYPT [SERVER_ENCRYPT; KERBEROS_ENCRYPT]

If **authentication** is SERVER_ENCRYPT, then the user ID and password are sent from the client to the server so authentication can take place on the server. User IDs and passwords sent over the network are encrypted.

A value of KERBEROS_ENCRYPT means that authentication is performed at a Kerberos server using the Kerberos security protocol for authentication.

Note: The KERBEROS_ENCRYPT authentication type is only supported on servers running Windows.

This parameter can only be updated from a Version 9 command line processor (CLP).

contact_host - Location of contact list

This parameter specifies the location where the contact information used for notification by the Scheduler and the Health Monitor is stored.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Null [any valid Db2 administration server TCP/IP hostname]

The location is defined to be a Db2 administration server's TCP/IP hostname. Allowing **contact_host** to be located on a remote DAS provides support for sharing a contact list across multiple Db2 administration servers. If **contact_host** is not specified, the DAS assumes the contact information is local.

This parameter can only be updated from a Version 8 command line processor (CLP).

das_codepage - DAS code page

This parameter indicates the code page used by the Db2 administration server.

Important: The Db2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in Db2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “Db2 administration server (DAS) has been deprecated” at .

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Null [any valid Db2 code page]

If the parameter is null, then the default code page of the system is used. This parameter should be compatible with the locale of the local Db2 instances. Otherwise, the Db2 administration server cannot communicate with the Db2 instances.

This parameter can only be updated from a Version 8 command line processor (CLP).

das_territory - DAS territory

This parameter shows the territory used by the Db2 administration server.

Important: The Db2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in Db2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “Db2 administration server (DAS) has been deprecated” at .

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Null [any valid Db2 territory]

If the parameter is null, then the default territory of the system is used.

This parameter can only be updated from a Version 8 command line processor (CLP).

dasadm_group - DAS administration authority group name

This parameter defines the group name with DAS Administration (DASADM) authority for the DAS.

Important: The Db2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in Db2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “Db2 administration server (DAS) has been deprecated” at .

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable

Default [range]

Null [any valid group name]

DASADM authority is the highest level of authority within the DAS.

DASADM authority is determined by the security facilities used in a specific operating environment.

- For the Windows operating systems, this parameter can be set to any local group that is defined in the Windows security database. Group names are accepted as long as they are 30 bytes or less in length. If “NULL” is specified for this parameter, all members of the Administrators group have DASADM authority.

- For Linux and UNIX systems, if “NULL” is specified as the value of this parameter, the DASADM group defaults to the primary group of the instance owner.
If the value is not “NULL”, the DASADM group can be any valid UNIX group name.

db2system - Name of the Db2 server system

This parameter specifies the name that is used by your users and database administrators to identify the Db2 server system.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable Online

Default [range]

TCP/IP host name [any valid system name]

If possible, this name should be unique within your network.

This name aids users in identifying the system that contains the database they want to access. A value for **db2system** is set at installation time as follows:

- On Windows, the **setup** program sets it equal to the computer name specified for the Windows system.
- On UNIX systems, it is set equal to the UNIX system's TCP/IP hostname.

discover - DAS discovery mode

This parameter determines the type of discovery mode that is started when the Db2 Administration Server starts.

Important: The Db2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in Db2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “Db2 administration server (DAS) has been deprecated” at .

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

SEARCH [DISABLE; KNOWN; SEARCH]

- If **discover** = SEARCH, the administration server handles SEARCH discovery requests from clients. SEARCH provides a superset of the functionality provided by KNOWN discovery. When **discover** = SEARCH, the administration server will handle both SEARCH and KNOWN discovery requests from clients.

- If **discover** = KNOWN, the administration server handles only KNOWN discovery requests from clients.
- If **discover** = DISABLE, then the administration server will not handle any type of discovery request. The information for this server system is essentially hidden from clients.

Discovery mode is enabled by default.

exec_exp_task - Execute expired tasks

This parameter specifies whether or not the Scheduler will execute tasks that have been scheduled in the past, but have not yet been executed.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable

Default [range]

No [Yes; No]

The Scheduler only detects expired tasks when it starts up. For example, if you have a job scheduled to run every Saturday, and the Scheduler is turned off on Friday and then restarted on Monday, the job scheduled for Saturday is now a job that is scheduled in the past. If **exec_exp_task** is set to Yes, your Saturday job will run when the Scheduler is restarted.

jdk_64_path - 64-Bit Software Developer's Kit for Java installation path DAS

This parameter specifies the directory under which the 64-Bit Software Developer's Kit (SDK) for Java, to be used for running Db2 administration server functions, is installed.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Null [any valid path]

Note: This is different from the **jdk_path** configuration parameter, which specifies a 32-bit SDK for Java.

Environment variables used by the Java interpreter are computed from the value of this parameter. This parameter is only used on those platforms that support both 32- and 64-bit instances.

Those platforms are:

- 64-bit kernels of AIX, HP-UX, and Solaris operating systems
- 64-bit Windows on x64 and IPF
- 64-bit Linux kernel on AMD64 and Intel EM64T systems (x64), POWER®, and zSeries.

On all other platforms, only **jdk_path** is used.

Because there is no default value for this parameter, you should specify a value when you install the SDK for Java.

This parameter can only be updated from a Version 8 command line processor (CLP).

jdk_path - Software Developer's Kit for Java installation path DAS

This parameter specifies the directory under which the Software Developer's Kit (SDK) for Java, to be used for running Db2 administration server functions, is installed.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable Online

Propagation class

Immediate

Default [range]

Default Java install path [any valid path]

Environment variables used by the Java interpreter are computed from the value of this parameter.

On Windows operating systems, Java files (if needed) are placed under the `sqllib` directory (in `java\jdk`) during Db2 installation. The **jdk_path** configuration parameter is then set to `sqllib\java\jdk`. Java is never actually installed by Db2 on Windows platforms; the files are merely placed under the `sqllib` directory, and this is done regardless of whether or not Java is already installed.

This parameter can only be updated from a Version 8 command line processor (CLP).

sched_enable - Scheduler mode

This parameter indicates whether or not the Scheduler is started by the administration server.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable

Default [range]
Off [On; Off]

The Scheduler allows tools such as the IBM Data Studio to schedule and execute tasks at the administration server.

This parameter can only be updated from a Version 8 command line processor (CLP).

sched_userid - Scheduler user ID

This parameter specifies the user ID used by the Scheduler to connect to the tools catalog database. This parameter is only relevant if the tools catalog database is remote to the Db2 administration server.

Configuration type
Db2 Administration Server

Applies to
Db2 Administration Server

Parameter type
Informational

Default [range]
Null [any valid user ID]

The userid and password used by the Scheduler to connect to the remote tools catalog database are specified using the **db2admin** command.

smtp_server - SMTP server

When the Scheduler is on, this parameter identifies the SMTP server that the Scheduler will use to send email and pager notifications.

Configuration type
Db2 Administration Server

Applies to
Db2 Administration Server

Parameter type
Configurable Online

Propagation class
Immediate

Default [range]
Null [any valid SMTP server TCP/IP hostname]

This parameter is used by the Scheduler and the Health Monitor.

This parameter can only be updated from a Version 8 command line processor (CLP).

toolscat_db - Tools catalog database

This parameter indicates the tools catalog database used by the Scheduler.

Configuration type
Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable

Default [range]

Null [any valid database alias]

This database must be in the database directory of the instance specified by **toolscat_inst**.

toolscat_inst - Tools catalog database instance

This parameter indicates the instance name that is used by the Scheduler, along with **toolscat_db** and **toolscat_schema**, to identify the tools catalog database.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable

Default [range]

Null [any valid instance]

The tools catalog database contains task information. The tools catalog database must be listed in the database directory of the instance specified by this configuration parameter. The database can be local or remote. If the tools catalog database is local, the instance must be configured for TCP/IP. If the database is remote, the database partition cataloged in the database directory must be a TCP/IP node.

toolscat_schema - Tools catalog database schema

This parameter indicates the schema of the tools catalog database used by the Scheduler.

Configuration type

Db2 Administration Server

Applies to

Db2 Administration Server

Parameter type

Configurable

Default [range]

Null [any valid schema]

The schema is used to uniquely identify a set of tools catalog tables and views within the database.

This parameter can only be updated from a Version 8 command line processor (CLP).

Ingest utility configuration parameters

You can set these configuration parameters to control how the INGEST utility performs on your Db2 client.

commit_count - Commit count configuration parameter

This parameter specifies the number of rows each flusher writes in a single transaction before issuing a commit.

Configuration type
Ingest utility

Applies to
Ingest utility

Parameter type
Configurable

Default [range]
0 [0 to max 32-bit signed integer]

If **commit_count** is not a multiple of 1,000, it is rounded to the nearest multiple and the ingest utility issues a warning (SQL2903W).

When **commit_count** is set to 0 (the default), **commit_period** is used, meaning that by default, the ingest utility commits transactions based on elapsed time only. If you want to commit transactions based on the number of rows only, you must set **commit_count** to a non-zero value and set **commit_period** to 0.

When neither **commit_count** nor **commit_period** is specified, the **commit_period** default setting of 1 second is used.

If both **commit_count** and **commit_period** are specified, the ingest utility honors both; that is, it issues a commit when it has written the specified number of rows or if there has not been a commit within the specified number of seconds.

Recommendations

If the row size is small, set **commit_count** to a large value. If the row size is large, set **commit_count** to a small value.

If no other applications are running, the absolute maximum commit count can be estimated very roughly using the following formula:

$$\frac{(\text{logfilsiz} * (\text{logprimary} + \text{logsecond}) * 4\text{KB})}{\text{divided by (total number of flushers)}}$$
 divided by (estimated row size + overhead)

If other applications are running, the maximum commit count is smaller.

If **commit_count** is set to too large a value, it is likely the lock list or the transaction log will fill up. In that case, the INGEST command terminates with error SQL0912N or SQL0964C. If you get SQL0912N or SQL0964C and you have the **retry_count** configuration parameter set, the ingest utility does both of the following

- adjusts the setting of either **commit_count** or **commit_period**, or both and issues a warning message
- issues a commit and retries the operation

commit_period - Commit period configuration parameter

Specifies the number of seconds between committed transactions.

Configuration type
Ingest utility

Applies to
Ingest utility

Parameter type
Configurable

Default [range]
1 [0 to 2,678,400 (31 days)]

Unit of measure
Seconds

At each flush, the ingest utility checks the number of seconds since the last commit. If it is greater than or equal to the setting of **commit_period**, the ingest utility issues a commit.

If **commit_period** is set to 0, the **commit_count** configuration parameter is used, meaning that transactions are to be committed based upon the number of rows.

When neither **commit_count** nor **commit_period** is specified, the **commit_period** default setting of 1 second is used.

If both **commit_count** and **commit_period** are specified, the ingest utility honors both; that is, it issues a commit when it has written the specified number of rows or if there has not been a commit within the specified number of seconds.

Recommendations

If the row size is small, set **commit_period** to a large value. If the row size is large, set **commit_period** to a small value.

If **commit_period** is set to too large a value, it is likely the lock list or the transaction log will fill up. In that case, the INGEST command terminates with error SQL0912N or SQL0964C. If you get SQL0912N or SQL0964C and you have the **retry_count** configuration parameter set, the ingest utility does both of the following actions

- adjusts the setting of either **commit_count** or **commit_period**, or both and issues a warning message
- issues a commit and retries the operation

num_flushers_per_partition - Number of flushers per database partition configuration parameter

Specifies the number of flushers to allocate for each database partition.

Configuration type
Ingest utility

Applies to
Ingest utility

Parameter type
Configurable

Default [range]

max(1, ((number of logical CPUs)/2)/(number of partitions)) [0 to system resources]

When **num_flushers_per_partition** is set to 0, one flusher is used for all partitions.

Note: When the operation is DELETE, MERGE, or UPDATE, the utility will sometimes reduce this parameter to 0 or 1 in order to reduce the chances of a deadlock occurring (SQL2903W).

num_formatters - Number of formatters configuration parameter

Specifies the number of formatters to allocate.

Configuration type

Ingest utility

Applies to

Ingest utility

Parameter type

Configurable

Default [range]

max(1, ((number of logical CPUs)/2)) [1 to system resources]

Although the default is the optimal setting, if you specify the DUMPFIL (or BADFILE) parameter on the **INGEST** command and you want the ingest utility to write the bad records in the same order that they appear in the input file, you must set **num_formatters** to 1.

pipe_timeout - Pipe timeout configuration parameter

This parameter specifies the maximum number of seconds to wait for data when the input source is a pipe.

Configuration type

Ingest command

Applies to

Ingest utility

Parameter type

Configurable

Default [range]

600 seconds (10 minutes) [0 to 2,678,400 (31 days)]

Unit of measure

seconds

When **pipe_timeout** is set to 0, the ingest utility waits indefinitely. If no data arrives within the specified period, the **INGEST** command ends and an error is returned.

retry_count - Retry count configuration parameter

Specifies the number of times to retry a failed, but recoverable, transaction.

Configuration type

Ingest utility

Applies to
Ingest utility

Parameter type
Configurable

Default [range]
0 [0 to 1,000]

The ingest utility only retries transactions that fail for one of the following reasons:

- A connection failed but has been reestablished.
- Deadlock or timeout with automatic rollback occurred.
- A system error has caused the unit of work to be rolled back.
- Virtual storage or database resource is not available.

retry_period - Retry period configuration parameter

Specifies the number of seconds to wait before retrying a failed, but recoverable, transaction.

Configuration type
Ingest utility

Applies to
Ingest utility

Parameter type
Configurable

Default [range]
0 [0 to 2,678,400 (31 days)]

Unit of measure
Seconds

The ingest utility only retries transactions that fail for one of the following reasons:

- A connection failed but has been reestablished.
- Deadlock or timeout with automatic rollback occurred.
- A system error has caused the unit of work to be rolled back.
- Virtual storage or database resource is not available.

shm_max_size - Maximum size of shared memory configuration parameter

Specifies the maximum size of Inter Process Communication (IPC) shared memory in bytes. Because the ingest utility runs on the client, this memory is allocated on the client machine.

Configuration type
Ingest command

Applies to
Ingest utility

Parameter type
Configurable

Default [range]
512 MB (4,294,967,296) [*n* to available memory]

n is calculated as follows:

$$\begin{aligned}
&11000 + \\
&(numTransporters \times 500) + \\
&(NUM_FORMATTERS \times 500) + \\
&(numUsedPartitions \times 50) + \\
&(totalNumFlushers \times 4000) + \\
&(MSG_BUF_COUNT \times (100 + MSG_BUF_SIZE)) + \\
&(numFields \times 66300) + \\
&(1.5 \times NUM_FORMATTERS \times sumOfAllFieldLengths)
\end{aligned}$$

where

- *numTransporters* is the number of input sources (if operation is INSERT or REPLACE), or 1 otherwise.
- *numUsedPartitions* is number of database partitions that the target table uses.
- *totalNumFlushers* is **num_flushers_per_partition** \times *numUsedPartitions*.
- *sumOfAllFieldLengths* is the total number of bytes in all the field definitions.
- *numFields* is the number of field definitions.

Unit of measure

bytes

Index

A

- agent_stack_sz database manager configuration parameter 27
- agentpri database manager configuration parameter 29
- agents
 - configuration parameters
 - affecting number of agents 21
 - agent_stack_sz 27
 - agentpri 29
 - applheapsz 120
 - aslheapsz 33
 - maxagents 80
 - maxcagents 81
 - num_poolagents 86
- alt_collate configuration parameter 114
- alt_diagpath database manager configuration parameter details 30
- alternate_auth_enc configuration parameter details 32
- app_ctl_heap_sz database configuration parameter 115
- appgroup_mem_sz database manager configuration parameter 116
- appl_memory database configuration parameter details 118
- applheapsz configuration parameter details 120
- application control heap size configuration parameter 115
- application support layer heap size configuration parameter 33
- applications
 - control heap 115
 - maximum number of coordinating agents at node 78
- archretrydelay configuration parameter 121
- aslheapsz configuration parameter 33
- audit_buf_sz configuration parameter details 34
- authentication
 - trust all clients configuration parameter 109
 - trusted clients authentication configuration parameter 110
- authentication configuration parameter 35
- authentication DAS configuration parameter 257
- authorities
 - defining group names
 - system administration authority group name configuration parameter 105
 - system control authority group name configuration parameter 106

- authorities (*continued*)
 - defining group names (*continued*)
 - system maintenance authority group name configuration parameter 106
- auto restart enable configuration parameter 124
- auto_del_rec_obj database configuration parameter 121
- auto_maint configuration parameter 122
- auto_reval database configuration parameter details 123
- autorestart database configuration parameter 124
- avg_appls configuration parameter 125

B

- backup_pending configuration parameter 126
- backups
 - tracking modified pages 250
- binding
 - configuration parameters 1, 2
- blk_log_dsk_ful configuration parameter details 126
- blocknonlogged database configuration parameter details 127
- buffer pools
 - query optimization 22
- bypass federated authentication configuration parameter 62

C

- catalog cache size configuration parameter 132
- catalog_noauth configuration parameter 42
- catalogcache_sz database configuration parameter 132
- CF self-tuning memory database configuration parameter details 128
- cf_catchup_trgt database configuration parameter 128
- cf_db_mem_sz database configuration parameter 129
- cf_diaglevel database manager configuration parameter 37
- cf_diagpath database manager configuration parameter details 38
- cf_gbp_sz database configuration parameter 130
- cf_lock_sz database configuration parameter 130

- cf_mem_sz database manager configuration parameter 39
- cf_num_conns database manager configuration parameter 39
- cf_num_workers database manager configuration parameter 40
- cf_sca_sz database configuration parameter 131
- cf_transport_method
 - details 41
- chnngpgs_thresh configuration parameter 133
- client I/O block size configuration parameter 91
- clients
 - client I/O block size configuration parameter 91
 - TCP/IP service name configuration parameter 104
- clnt_krb_plugin configuration parameter 42
- clnt_pw_plugin configuration parameter 43
- cluster_mgr database manager configuration parameter 43
- clusters
 - managing
 - cluster manager name configuration parameter 43
- code pages
 - database configuration parameter 134
- codepage database configuration parameter 134
- codeset database configuration parameter 134
- collate_info database configuration parameter 134
- comm_bandwidth database manager configuration parameter details 44
 - query optimization 22
- COMM_EXIT_LIST configuration parameter 44
- commit_count configuration parameter details 265
- commit_period configuration parameter 266
- commits
 - mincommit configuration parameter 198
- communications
 - connection elapse time configuration parameter 45
- concurrency
 - maximum number of active applications 193
- configuration file release level configuration parameter 89
- configuration files
 - details 1

- configuration parameters
 - agent_stack_sz 27
 - agentpri 29
 - agents 21
 - alt_collate 114
 - alt_diagpath 30
 - alternate_auth_enc 32
 - app_ctl_heap_sz 115
 - appgroup_mem_sz 116
 - appl_memory 118
 - applheapsz 120
 - archretrydelay 121
 - aslheapsz 33
 - audit_buf_sz 34
 - authentication 35
 - authentication (DAS) 257
 - auto_del_rec_obj 121
 - auto_maint 122
 - auto_reval 123
 - autorestart 124
 - avg_appls 125
 - backup_pending 126
 - blk_log_dsk_ful 126
 - blocknonlogged 127
 - catalog_noauth 42
 - catalogcache_sz 132
 - CF self-tuning memory 128
 - cf_catchup_trgt 128
 - cf_db_mem_sz 129
 - cf_diaglevel 37
 - cf_diagpath 38
 - cf_gbp_sz 130
 - cf_lock_sz 130
 - cf_mem_sz 39
 - cf_num_conns 39
 - cf_num_workers 40
 - cf_sca_sz 131
 - cf_transport_method 41
 - chnpgs_thresh 133
 - clnt_krb_plugin 42
 - clnt_pw_plugin 43
 - cluster_mgr 43
 - codepage 134
 - codeset 134
 - collate_info 134
 - comm_bandwidth 44
 - comm_exit_list 44
 - commit_count 265
 - commit_period 266
 - configuring Db2 database manager 2
 - conn_elapse 45
 - connect_proc 135
 - contact_host 258
 - cpuspeed 45
 - cur_commit 137
 - cur_eff_arch_level 46
 - cur_eff_code_level 46
 - DAS configuration parameters 257
 - das_codepage 258
 - das_territory 259
 - dasadm_group 259
 - database
 - changing values 1
 - database configuration parameters 114
 - database manager configuration parameters 27
- configuration parameters (*continued*)
 - database_consistent 137
 - database_level 138
 - database_memory 138
 - date_compat 47, 145
 - db_mem_thresh 144
 - Db2 Administration Server
 - configuration parameters 257
 - db2system 260
 - dbheap 142
 - dec_arithmetic - DECIMAL arithmetic mode 146
 - dec_to_char_fmt 147
 - decflt_rounding 148
 - details 1
 - dft_account_str 47
 - dft_degree 150
 - dft_extent_sz 151
 - dft_loadrec_ses 151
 - dft_monswitches 48
 - dft_mttb_types 152
 - dft_prefetch_sz 152
 - dft_queryopt 154
 - dft_refresh_age 154
 - dft_schemas_dcc 155
 - dft_sqlmathwarn 155
 - dft_table_org 157
 - dftdbpath 49
 - diaglevel 50
 - diagpath 51
 - diagsize 55
 - dir_cache 56
 - discover 58
 - discover (DAS) 260
 - discover_db 158
 - discover_inst 58
 - dlchktime 158
 - enable_xmlchar 159
 - enclib - Encryption library 159
 - encropts - Encryption options 160
 - encrypted_database 74
 - exec_exp_task 261
 - extended_row_sz 160
 - failarchpath 161
 - fcm_buffer_size 59
 - fcm_num_buffers 59
 - fcm_num_channels 60
 - fcm_parallelism 61
 - fed_noauth 62
 - federated 62
 - federated_async 63
 - fenced_pool 63
 - group_plugin 65
 - groupheap_ratio 161
 - hadr_db_role 162
 - hadr_local_host 162
 - hadr_local_svc 163
 - hadr_peer_window
 - details 164
 - hadr_remote_host 165
 - hadr_remote_inst 165
 - hadr_remote_svc 166
 - hadr_replay_delay 167
 - hadr_spool_limit 168
- configuration parameters (*continued*)
 - HADR_SSL_LABEL - Set up SSL communication between primary and standby HADR instances parameter 169
 - hadr_syncmode 170
 - hadr_target_list 172
 - hadr_timeout 174
 - health_mon 65
 - indexrec 66, 174
 - ingest utility 5
 - ingest utility configuration parameters 265
 - instance_memory 68
 - intra_parallel 71
 - java_heap_sz 72
 - jdk_64_path 261
 - jdk_path 73
 - jdk_path (DAS) 262
 - keepfenced 73
 - keystore_location 74
 - keystore_type 75
 - local_gssplugin 76
 - locklist 176
 - locktimeout 179
 - log_appl_info 180
 - log_ddl_stmts 180
 - log_retain_status 180
 - logarchcompr1 181
 - logarchcompr2 181
 - logarchmeth1 182
 - logarchmeth2 183
 - logarchopt1
 - details 185
 - logarchopt2 186
 - logbufsz 186
 - logfilsiz 187
 - loghead 188
 - logindexbuild 188
 - logpath 189
 - logprimary 189
 - logsecond 190
 - max_connections
 - details 76
 - restrictions 24
 - max_connretries 77
 - max_coordagents
 - details 78
 - restrictions 24
 - max_querydegree 79
 - max_time_diff 80
 - maxagents 80
 - maxappls 193
 - maxcagents 81
 - maxfilop 194
 - maxlocks 195
 - maxlog 192
 - min_dec_div_3 196
 - mincommit 198
 - mirrorlogpath 199
 - mon_act_metrics 201
 - mon_deadlock 202
 - mon_heap_sz 82
 - mon_lck_msg_lvl 205
 - mon_locktimeout 203
 - mon_lockwait 204
 - mon_lw_thresh 205

configuration parameters (*continued*)

mon_obj_metrics 206
 mon_pkglst_sz 208
 mon_req_metrics 208
 mon_rtn_data 209
 mon_rtn_execlist 210
 mon_uow_data 211
 mon_uow_execlist 212
 mon_uow_pkglst 212
 multipage_alloc 213
 nchar_mapping 213
 newlogpath 214
 nodetype 83
 notifylevel 84
 num_db_backups 215
 num_flushers_per_partition 266
 num_formatters 267
 num_freqvalues 216
 num_initagents 85
 num_initfenced 85
 num_iocleaners 217
 num_ioservers 218
 num_poolagents 86
 num_quantiles 220
 numarchretry 221
 number_compat 222
 numdb 87
 numlogspan 219
 numsegs 222
 opt_direct_wrkld 222
 overflowlogpath 223
 page_age_trgt_gcr 224
 page_age_trgt_mcr 225
 pagesize 225
 pckcachesz 225
 pipe_timeout 267
 PL_STACK_TRACE 227
 priv_mem_thresh 228
 query_optimization 22
 query_heap_sz 88
 rec_his_retentn 228
 recompiling query after configuration changes 24
 release 89
 restore_pending 229
 restrict_access 229
 resync_interval 91
 retry_count 267
 retry_period 268
 rollfwd_pending 230
 rqrioblk 91
 rstrt_light_mem 89
 sched_enable 262
 sched_userid 263
 section_actuals 230
 self_tuning_mem 231
 seqdetect 232
 sheapthres 92
 sheapthres_shr 233
 shm_max_size 268
 smtp_server 237, 263
 softmax 237
 sortheap 239
 spm_log_file_sz 94
 spm_log_path 95
 spm_max_resync 95
 spm_name 96

configuration parameters (*continued*)

sql_cclflags 243
 srv_plugin_mode 98
 srvcon_auth 96
 srvcon_gssplugin_list 97
 srvcon_pw_plugin 97
 ssl_cipherspecs 98
 ssl_clnt_keydb 99
 ssl_clnt_stash 100
 ssl_svcname 103
 ssl_svr_keydb 101
 ssl_svr_label 101
 ssl_svr_stash 102
 ssl_versions 104
 start_stop_time 102
 stat_heap_sz 244
 stmt_conc 244
 stmtheap 246
 string_units 247
 summary 5
 suspend_io 248
 svcname 104
 sysadm_group 105
 sysctrl_group 106
 sysmaint_group 106
 sysmon_group 107
 systime_period_adj 248
 territory 249
 tm_database 107
 toolscat_db 263
 toolscat_inst 264
 toolscat_schema 264
 tp_mon_name 108
 trackmod 250
 trust_allclnts 109
 trust_clntauth 110
 tsm_mgmtclass 250
 tsm_nodename 251
 tsm_owner 251
 tsm_password 251
 user_exit_status 252
 util_heap_sz 252
 util_impact_lim 110
 varchar2_compat 254
 vendoropt details 254
 WLM_ADMISSION_CTRL - WLM admission control 254
 WLM_AGENT_LOAD_TRGT - WLM agent load target 255
 wlm_collect_int 255
 WLM_CPU_LIMIT - WLM CPU limit 256
 WLM_CPU_SHARE_MODE - WLM CPU share mode 257
 WLM_CPU_SHARES - WLM CPU shares 256
 wlm_disp_concur 112
 wlm_disp_cpu_shares 113
 wlm_disp_min_util 114
 wlm_dispatcher 111
 conn_elapse configuration parameter 45
 connect stored procedure name configuration parameter 135
 connect_proc configuration parameter 135

connection elapsed time configuration parameter 45

connections elapsed time 45
 contact_host configuration parameter 258
 Coordinated Universal Time max_time_diff configuration parameter 80
 cpuspeed configuration parameter details 45
 query optimization effect 22
 cur_commit database configuration parameter details 137
 cur_eff_arch_level database manager configuration parameter 46
 cur_eff_code_level database manager configuration parameter 46

D

DAS configuration parameters 257
 DAS discovery mode configuration parameter 260
 das_codepage configuration parameter 258
 das_territory configuration parameter 259
 dasadm_group configuration parameter 259
 database configuration parameters overview 114
 See also configuration parameters 5 summary 5
 database encryption state 74
 database heap configuration parameter 142
 database manager machine node type configuration parameter 83
 starting 102
 stopping 102
 database manager configuration parameters overview 27
 See also configuration parameters 5 summary 5 update 26
 database system monitor default database system monitor switches configuration parameter 48
 database territory code configuration parameter 136
 database_consistent configuration parameter 137
 database_level configuration parameter 138
 database_memory database configuration parameter details 138
 databases appl_memory configuration parameter 118
 autorestart configuration parameter 124

- databases (*continued*)
 - backup_pending configuration parameter 126
 - codepage configuration parameter 134
 - codeset configuration parameter 134
 - collating information 134
 - maximum number of concurrently active databases configuration parameter 87
 - release level configuration parameter 89
 - territory code configuration parameter 136
 - territory configuration parameter 249
 - date_compat database configuration parameter
 - details 47, 145
 - db_mem_thresh configuration parameter 144
 - Db2 administration server (DAS) configuration parameters
 - authentication 257
 - contact_host 258
 - das_codepage 258
 - das_territory 259
 - dasadm_group 259
 - db2system 260
 - exec_exp_task 261
 - jdk_64_path 261
 - jdk_path 262
 - sched_enable 262
 - sched_userid 263
 - smtp_server 263
 - toolscat_db 263
 - toolscat_inst 264
 - toolscat_schema 264
 - Db2 Administration Server configuration parameters 257
 - DB2INSTPROF registry variable
 - location 1
 - db2system configuration parameter 260
 - dbheap database configuration parameter
 - details 142
 - deadlocks
 - checking for 158
 - dlchktime configuration parameter 158
 - dec_arithmetic - DECIMAL arithmetic mode configuration parameter 146
 - dec_to_char_fmt database configuration parameter
 - details 147
 - decflt_rounding database configuration parameter 148
 - decimal division scale to 3 configuration parameter 196
 - default database path configuration parameter 49
 - default number of SMS containers configuration parameter 222
 - default string units configuration parameter 247
 - dft_account_str configuration parameter 47
 - dft_degree configuration parameter
 - details 150
 - dft_degree configuration parameter (*continued*)
 - effect on query optimization 22
 - dft_extent_sz configuration parameter 151
 - dft_loadrec_ses configuration parameter 151
 - dft_mon_bufpool configuration parameter 48
 - dft_mon_lock configuration parameter 48
 - dft_mon_sort configuration parameter 48
 - dft_mon_stmt configuration parameter 48
 - dft_mon_table configuration parameter 48
 - dft_mon_timestamp configuration parameter 48
 - dft_mon_uow configuration parameter 48
 - dft_monswitches configuration parameter 48
 - dft_mttb_types configuration parameter 152
 - dft_prefetch_sz configuration parameter 152
 - dft_queryopt configuration parameter 154
 - dft_refresh_age configuration parameter 154
 - dft_schemas_dcc configuration parameter
 - details 155
 - dft_sqlmathwarn configuration parameter 155
 - dft_table_org database configuration parameter
 - details 157
 - dfdtbpath configuration parameter 49
 - diaglevel configuration parameter
 - details 50
 - diagnostic data directory path 30, 51
 - diagpath database manager configuration parameter
 - details 51
 - diagsize database manager configuration parameter
 - details 55
 - dir_cache configuration parameter 56
 - directory cache support configuration parameter
 - details 56
 - discover server instance configuration parameter 58
 - discover_db configuration parameter 158
 - discover_inst configuration parameter 58
 - discovery feature
 - discovery mode configuration parameter 58
 - discovery mode configuration parameter 58
 - dlchktime configuration parameter 158
- ## E
- enable_xmlchar database configuration parameter 159
 - encrlib - Encryption library configuration parameter 159
 - encropts - Encryption options configuration parameter 160
 - encrypted_database configuration parameter
 - details 74
 - exec_exp_task configuration parameter 261
 - extended_row_sz database configuration parameter
 - details 160
- ## F
- failarchpath configuration parameter 161
 - FCM
 - channels 60
 - configuration parameters
 - fcm_buffer_size 59
 - fcm_num_buffers 59
 - fcm_num_channels 60
 - fcm_buffer_size configuration parameter
 - details 59
 - fcm_num_buffers configuration parameter
 - details 59
 - fcm_num_channels configuration parameter
 - details 60
 - fcm_parallelism configuration parameter 61
 - fed_noauth configuration parameter 62
 - federated configuration parameter 62
 - federated databases
 - system support configuration parameter 62
 - federated_async database manager configuration parameter 63
 - fenced_pool database manager configuration parameter 63
 - first active log file configuration parameter 188
- ## G
- group_plugin configuration parameter 65
 - groupheap_ratio database manager configuration parameter 161
- ## H
- hadr_db_role configuration parameter 162
 - hadr_local_host configuration parameter 162
 - hadr_local_svc configuration parameter 163

hadr_peer_window database configuration parameter details 164
 hadr_remote_host configuration parameter details 165
 hadr_remote_inst configuration parameter details 165
 hadr_remote_svc configuration parameter details 166
 hadr_replay_delay database configuration parameter details 167
 hadr_spool_limit database configuration parameter details 168
 HADR_SSL_LABEL - Set up SSL communication between primary and standby HADR instances parameter configuration parameter 169
 hadr_syncmode database configuration parameter details 170
 hadr_target_list configuration parameter 172
 hadr_timeout database configuration parameter details 174
 health monitor health_mon configuration parameter 65
 health_mon configuration parameter 65
I
 indexrec configuration parameter 66, 174
 ingest utility configuration parameters commit_count 265 commit_period 266 num_flushers_per_partition 266 num_formatters 267 pipe_timeout 267 retry_count 267 retry_period 268 shm_max_size 268
 ingest utility configuration parameters 265
 initial number of agents in pool configuration parameter 85
 initial number of fenced processes configuration parameter 85
 instances instance_memory configuration parameter 68
 intra_parallel database manager configuration parameter 71
J
 java_heap_sz database manager configuration parameter 72
 jdk_64_path configuration parameter 261

jdk_path configuration parameter details 73
 jdk_path DAS configuration parameter 262
K
 keepfenced configuration parameter details 73
 keystore location 74
 keystore type 75
 keystore_location configuration parameter details 74
 keystore_type configuration parameter details 75

L
 local_gssplugin configuration parameter 76
 locklist configuration parameter details 176 query optimization 22
 locks maximum percent of lock list before escalation configuration parameter 195 maximum storage for lock list configuration parameter 176 time interval for checking deadlock configuration parameter 158
 locktimeout configuration parameter 179
 log_appl_info configuration parameter details 180
 log_ddl_stmts configuration parameter details 180
 log_retain_status configuration parameter 180
 logarchcompr1 configuration parameter details 181
 logarchcompr2 configuration parameter details 181
 logarchmeth1 configuration parameter details 182
 logarchmeth2 configuration parameter details 183
 logarchopt1 configuration parameter details 185
 logarchopt2 configuration parameter details 186
 logbufsz database configuration parameter details 186
 logfilesiz database configuration parameter details 187
 loghead configuration parameter 188
 logindexbuild configuration parameter 188
 logpath configuration parameter 189
 logprimary database configuration parameter details 189
 logs configuration parameters blk_log_dsk_ful 126

logs (continued) configuration parameters (continued) log_retain_status 180 logbufsz 186 logfilesiz 187 loghead 188 logpath 189 logprimary 189 logsecond 190 mirrorlogpath 199 newlogpath 214 overflowlogpath 223 softmax 237
 logsecond configuration parameter overview 190
M
 max_connections database manager configuration parameter 24, 76
 max_connretries configuration parameter 77
 max_coordagents database manager configuration parameter details 78 restrictions 24
 max_querydegree configuration parameter 79
 max_time_diff database manager configuration parameter details 80
 maxagents database manager configuration parameter details 80
 maxappls configuration parameter details 193
 maxcagents database manager configuration parameter 81
 MAXDARI configuration parameter renamed to fenced_pool configuration parameter 63
 maxfilop database configuration parameter 194
 maximum database files open per application configuration parameter 194
 maximum Java interpreter heap size configuration parameter 72
 maximum number of active applications configuration parameter 193
 maximum number of agents configuration parameter 80
 maximum number of concurrent agents configuration parameter 81
 maximum number of concurrently active databases configuration parameter 87
 maximum number of coordinating agents configuration parameter 78
 maximum number of fenced processes configuration parameter 63
 maximum percentage of lock list before escalation configuration parameter 195
 maximum percentage of log per transaction configuration parameter 192

- maximum query degree of parallelism configuration parameter
 - details 79
 - effect on query optimization 22
- maximum size of application group memory set configuration parameter 116
- maximum storage for lock list configuration parameter 176
- maximum time difference between members configuration parameter 80
- maxlocks configuration parameter details 195
- maxlog configuration parameter 192
- members
 - maximum time difference among 80
- memory
 - apmheapsz configuration parameter 120
 - application memory configuration parameter 118
 - aslheapsz configuration parameter 33
 - dbheap configuration parameter 142
 - instance memory configuration parameter 68
 - package cache size configuration parameter 225
 - sort heap size configuration parameter 239
 - sort heap threshold configuration parameter 92
 - statement heap size configuration parameter 246
- min_dec_div_3 configuration parameter 196
- mincommit database configuration parameter
 - details 198
- mirrorlogpath database configuration parameter
 - details 199
- mon_act_metrics configuration parameter details 201
- mon_deadlock configuration parameter details 202
- mon_heap_sz database manager configuration parameter details 82
- mon_lck_msg_lvl configuration parameter 205
- mon_locktimeout configuration parameter 203
- mon_lockwait configuration parameter details 204
- mon_lw_thresh configuration parameter details 205
- mon_obj_metrics database configuration parameter details 206
- mon_pkglist_sz configuration parameter 208
- mon_req_metrics configuration parameter details 208
- mon_rtn_execlist database configuration parameter
 - details 210

- mon_uow_data database configuration parameter
 - details 211
- mon_uow_execlist database configuration parameter
 - details 212
- mon_uow_pkglist database configuration parameter
 - details 212
- monitoring
 - mon_rtn_execlist database configuration parameter 210
 - monitoring routine capture configuration parameter 209
 - monitoring routine capture database configuration parameter 209
 - monitoring routine executable list configuration parameter 210
 - multipage_alloc configuration parameter 213

N

- national character string mapping configuration parameter 213
- nchar_mapping configuration parameter 213
- newlogpath database configuration parameter
 - details 214
- node connection retries configuration parameter 77
- nodes
 - connection elapsed time configuration parameter 45
 - maximum coordinating agents configuration parameter 78
- nodetype configuration parameter 83
- notify level configuration parameter
 - overview 84
- num_db_backups configuration parameter 215
- num_flushers_per_partition configuration parameter 266
- num_formatters configuration parameter 267
- num_freqvalues configuration parameter 216
- num_initagents database manager configuration parameter
 - details 85
- num_initfenced database manager configuration parameter 85
- num_iocleaners configuration parameter 217
- num_ioservers configuration parameter 218
- num_log_span configuration parameter 219
- num_poolagents database manager configuration parameter
 - details 86
- num_quantiles configuration parameter 220

- numarchretry configuration parameter 221
- number log span configuration parameter 219
- number of commits to group configuration parameter 198
- number of database backups configuration parameter 215
- number_compat database configuration parameter
 - details 222
- numdb database manager configuration parameter
 - details 87
- numsegs database configuration parameter 222

O

- opt_direct_wrkld configuration parameter 222
- overflowlogpath database configuration parameter
 - details 223

P

- page_age_trgt_gcr configuration parameter 224
- page_age_trgt_mcr configuration parameter 225
- pages
 - sizes
 - database default 225
- pagesize configuration parameter 225
- parallelism
 - configuration parameters
 - dft_degree 150
 - intra_parallel 71
 - max_querydegree 79
- pckcachesz database configuration parameter
 - details 225
- pipe_timeout configuration parameter 267
- PL_STACK_TRACE configuration parameter 227
- pool size for agents configuration parameter 86
- priv_mem_thresh database manager configuration parameter 228
- protocols
 - TCP/IP service name configuration parameter 104

Q

- queries
 - statement heap size configuration parameter 246
- query optimization
 - configuration parameters 22
- query_heap_sz database manager configuration parameter 88

R

- rec_his_retentn configuration parameter 228
- recovery
 - auto restart enable configuration parameter 124
 - backup pending indicator configuration parameter 126
 - default number of load recovery sessions configuration parameter 151
 - index re-creation time configuration parameter 66, 174
 - log retain status indicator configuration parameter 180
 - number of database backups configuration parameter 215
 - restore pending configuration parameter 229
 - rollforward pending indicator configuration parameter 230
 - user exit status indicator configuration parameter 252
- recovery history file
 - retention period configuration parameter 228
- recovery range and soft checkpoint interval configuration parameter 237
- release configuration parameter 89
- restore_pending configuration parameter 229
- restrict_access configuration parameter 229
- RESTRICTIVE parameter of CREATE DATABASE command indicating use 229
- resync_interval configuration parameter 91
- retry_count configuration parameter 267
- retry_period configuration parameter 268
- rollforward utility
 - roll forward pending indicator 230
- rollfwd_pending configuration parameter 230
- routines
 - monitoring routine capture configuration parameter 209
 - monitoring routine executable list configuration parameter 210
- rqrioblk configuration parameter details 91
- rstrt_light_mem database manager configuration parameter details 89

S

- sched_enable configuration parameter 262
- sched_userid configuration parameter 263
- section_actuals configuration parameter 230

- security
 - plug-ins
 - configuration parameters 35, 42, 43, 96, 98
- self_tuning_mem configuration parameter 231
- seqdetect configuration parameter 232
- sheapthres configuration parameter 92
- sheapthres_shr configuration parameter 233
- shm_max_size configuration parameter 268
- smtp_server configuration parameter 263
- smtp_server database configuration parameter 237
- softmax database configuration parameter 237
- sortheap database configuration parameter
 - details 239
 - effect on query optimization 22
- sorting
 - sort heap size configuration parameter 239
 - sort heap threshold configuration parameter 92
 - sort heap threshold for shared sorts configuration parameter 233
- split mirrors
 - database I/O operations state configuration parameter 248
- spm_log_file_sz configuration parameter 94
- spm_log_path configuration parameter 95
- spm_max_resync configuration parameter 95
- spm_name configuration parameter 96
- SQL statements
 - statement heap size configuration parameter 246
- sql_ccflags database configuration parameter 243
- SQLDBCON database configuration file
 - configuring Db2 database manager 2
 - overview 1
- SQLDBCONF database configuration file
 - configuring Db2 database manager 2
 - overview 1
- srv_plugin_mode configuration parameter 98
- srvcon_auth configuration parameter 96
- srvcon_gssplugin_list configuration parameter 97
- srvcon_pw_plugin configuration parameter 97
- ssl_cipherspecs configuration parameter details 98
- ssl_clnt_keydb configuration parameter details 99
- ssl_clnt_stash configuration parameter details 100
- ssl_svcname configuration parameter details 103
- ssl_svr_keydb configuration parameter details 101

- ssl_svr_label configuration parameter 101
- ssl_svr_stash configuration parameter details 102
- ssl_versions configuration parameter details 104
- start and stop timeout configuration parameter 102
- start_stop_time configuration parameter 102
- stat_heap_sz database configuration parameter 244
- statement heap size configuration parameter 246
- STMM
 - enabling 231
- stmt_conc database configuration parameter 244
- stmthep database configuration parameter
 - details 246
 - effect on query optimization 22
- string_units configuration parameter 247
- suspend_io database configuration parameter 248
- svcname configuration parameter 104
- sysadm_group configuration parameter details 105
- sysctrl_group configuration parameter 106
- sysmaint_group configuration parameter 106
- sysmon_group configuration parameter 107
- systeme_period_adj configuration parameter 248

T

- TCP/IP service name configuration parameter 104
- territory configuration parameter 249
- time
 - maximum difference between members 80
- Tivoli Storage Manager
 - management class configuration parameter 250
 - node name configuration parameter 251
 - owner name configuration parameter 251
 - password configuration parameter 251
- tm_database configuration parameter 107
- toolscat_db configuration parameter 263
- toolscat_inst configuration parameter 264
- toolscat_schema configuration parameter 264
- tp_mon_name configuration parameter 108
- track modified pages configuration parameter 250
- trackmod configuration parameter 250

- transaction processing monitors
 - transaction processor monitor name configuration parameter 108
- trust_allclnts configuration parameter 109
- trust_clntauth configuration parameter 110
- tsm_mgmtclass configuration parameter 250
- tsm_nodename configuration parameter 251
- tsm_owner configuration parameter 251
- tsm_password configuration parameter 251

U

- update
 - database manager configuration parameters 26
- user exit status indicator configuration parameter 252
- user_exit_status configuration parameter 252
- util_heap_sz configuration parameter 252
- util_impact_lim configuration parameter 110

V

- varchar2_compat database configuration parameter
 - details 254
- vendoropt configuration parameter
 - details 254

W

- WLM_ADMISSION_CTRL - WLM admission control configuration parameter 254
- WLM_AGENT_LOAD_TRGT - WLM agent load target configuration parameter 255
- wlm_collect_int database configuration parameter 255
- WLM_CPU_LIMIT - WLM CPU limit configuration parameter 256
- WLM_CPU_SHARE_MODE - WLM CPU share mode configuration parameter 257
- WLM_CPU_SHARES - WLM CPU shares configuration parameter 256
- wlm_disp_concur configuration parameter
 - details 112
- wlm_disp_cpu_shares configuration parameter 113
- wlm_disp_min_util configuration parameter
 - details 114
- wlm_dispatcher configuration parameter
 - details 111
- workload management dispatcher configuration parameter 111

- workload manager dispatcher CPU shares configuration parameter 113
- workload manager dispatcher minimum CPU utilization configuration parameter 114
- workload manager dispatcher thread concurrency configuration parameter 112

X

- XQuery statements
 - optimization configuration parameters 22
 - statement heap size configuration parameter 246



Printed in USA