HCL OneDB 2.0.1

OneDB Administrator's Reference

Contents

Chapter 1. Administrator's Reference

The *HCL OneDB™ Administrator's Reference* includes comprehensive descriptions of HCL OneDB™ configuration parameters, the system-monitoring interface (SMI) tables in the sysmaster database, the syntax of database server utilities such as onmode and onstat, logical-log records, disk structures, event alarms, and unnumbered error messages.

These topics are of interest to the following users:

- Database administrators
- System administrators
- Performance engineers

These topics are written with the assumption that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with database server administration, operating-system administration, or network administration

Configuring and monitoring HCL OneDB™

Database configuration parameters

The HCL OneDB™ database server uses a configuration file, which is called the `onconfig` file, during initialization. This file contains default configuration parameter values. You can modify the parameter values to improve performance and other characteristics of the instance or database.

The **ONCONFIG** environment variable identifies your `onconfig` file.

onconfig file

When you add or change information in the `onconfig` file, you must follow the conventions that are used in the file.

The parameter description and the possible values are specified in the comments above their entries in the `onconfig.std` file.

The following line shows the syntax for a parameter line:

```
PARAMETER_NAME parameter_value comments
```

The following rules describe the `onconfig` file behavior:

- Each parameter is on a separate line.
- Lines that start with the `#` symbol are comments.
- The maximum line limit of the `onconfig` file is 512 bytes. Lines that exceed this limit are truncated and might cause configuration problems.

- White space (tabs, spaces, or both) is required between the parameter name, the parameter value, and an optional comment. Do not use any tabs or spaces within a parameter value. Any characters after the parameter value and blank space are interpreted as comments, regardless of whether they are preceded by a `#` symbol.
- Parameters and their values are case-sensitive. The parameter names are always uppercase. If the value entry is described with uppercase letters, you must use uppercase (for example, the CPU value of the NETTYPE parameter).
- Most parameters can have one valid entry. If more than one entry for these parameters exists in the `onconfig` file, the first entry is used. Some parameters, however, can have multiple entries, such as the DBSERVERALIASES configuration parameter, which requires a comma between entries. Some parameters, such as the VPCLASS configuration parameter, can exist multiple times.
- Unrecognized parameters are copied but ignored and no error is given.

Tip: If you run a utility like grep on the `onconfig.std` template file, specify the new line character (^) to return just the configuration parameter name and value. Without the new line character, the parameter description is also returned.

For example, the following command returns both the configuration parameter description and the value:

```
grep "MSGPATH" onconfig.std
# MSGPATH      - The path of the IDS message log file
MSGPATH $ONEDB_HOME/tmp/online.log
```

Whereas, the following command returns only the configuration parameter value:

```
grep "^MSGPATH" onconfig.std
MSGPATH $ONEDB_HOME/tmp/online.log
```

Conventions for environment variables

You can enter an environment variable as a value in any configuration parameter in which the variable is applicable. For example, for the DBSERVERNAME configuration parameter you can specify the following environment variable instead of the name of your database server:

```
DBSERVERNAME     $MY_DBSERVERNAME
```

Important: If you enter an environment variable as a value, you must set that environment variable in the environment of any executable program or utility that reads the `onconfig` file. Utilities that read the `onconfig` file include the oninit, oncheck, onbar, onlog, and archecker utilities.

Modifying the onconfig file

You can modify the `onconfig` file for your database server to customize server function or tune server behavior.

About this task

By default, the `onconfig` file is in the `ONEDB_HOME/etc` directory. The **ONCONFIG** environment variable specifies the name and location of the `onconfig` file.

The `onconfig.std` file is a template configuration file from which you can copy configuration parameter settings. The `onconfig.std` file is a template and not a functional configuration. You can copy and rename the `onconfig.std` file, but do not modify or delete the`onconfig.std` file. If you omit a parameter value in your copy of the configuration file, the database server either uses default values in `onconfig.std` template file or calculates values that are based on other parameter values.

You can modify the `onconfig` file by any of the following methods:

- You can use a text editor to modify configuration parameter values. The changes take effect after the next time the database server is shut down and restarted.
- You can modify the values of many configuration parameters dynamically without restarting the database server by running the onmode -wf to update configuration parameters permanently or by running the onmode -wm command to update configuration parameters in memory.
- You can generate an `onconfig` file with settings that are optimized for the connections, disk space, and CPU usage that you estimate by running the genoncfg utility.
- You can export, import, and modify configuration parameters in groups:
  - Use the onmode -we command to export a snapshot of the current configuration to a file. The resulting snapshot can then be archived, used as a configuration file, or imported to another running instance.
  - Use the onmode -wi command to import tunable configuration parameters from a previously exported file. Configuration parameters in the file that are not dynamically tunable are ignored.
- You can modify, reset, export, and import a configuration file with SQL administration API commands:
  - Use **modify config** argument with the admin() or task() function to change the value of a configuration parameter.
  - Use the **export config** and **import config** arguments with the admin() or task() function to export or import a file that contains one or more dynamically tunable configuration parameters.
  - Use the **reset config** or **reset config all** argument with the admin() or task() function to revert the value of a configuration parameter or all configuration parameters to its value in the `onconfig` file.

You can compare two `onconfig` files by running the onconfig_diff utility.

Displaying the settings in the `onconfig` file

There are several tools that you can use to display the settings in the `onconfig` file.

To display the settings in the `onconfig` file, use one of the following tools:

Choose from:

- Open the `onconfig` file with a text editor.
- View the contents of the `onconfig` file with the onstat -c command.

- View a list of configuration parameters and their current values by running the onstat -g cfg command. If configuration parameters are updated dynamically, the current values differ from the permanent values in the `onconfig` file.

  You can use additional options with the onstat -g cfg command to display only the configuration parameters that were changed dynamically or to display additional information about all configuration parameters.

onconfig Portal: Configuration parameters by functional category

The information in this section lists configuration parameters as they are in the UNIX™ `onconfig.std` file.

Category list

To use this section, you first determine the appropriate category from the following list, then follow the link to the configuration parameters for that category. The categories are listed in the same order as they are in the `onconfig.std` file. Parameters that are not in the `onconfig.std` file but that you can add to your `onconfig` file are listed in Table 57: Parameters that are not in the onconfig.std file on page 32.

Root dbspace configuration parameters

Use the following configuration parameters to configure the root dbspace.

Table 1. Root dbspace configuration parameters

Configuration Parameter	Reference
ROOTNAME configuration parameter on page 152	The root dbspace name.
ROOTPATH configuration parameter on page 153	The path for the root dbspace.
ROOTOFFSET configuration parameter on page 153	The offset for the root dbspace.
ROOTSIZE configuration parameter on page 154	The size of the root dbspace.
MIRROR configuration parameter on page 129	Enables or disables mirroring.
MIRRORPATH configuration parameter on page 130	The path for the mirrored root dbspace.
MIRROROFFSET configuration parameter on page 129	The offset for the mirrored root dbspace.

Physical log configuration parameters

Use the following configuration parameters to configure physical logs.

Table 2. Physical log configuration parameters

Configuration Parameter	Reference
PHYSFILE configuration parameter on page 144	The size of the physical log.
PLOG_OVERFLOW_PATH configuration parameter on page 145	The overflow directory for physical log files.
PHYSBUFF configuration parameter on page 143	The size of the physical log buffer.

Logical log configuration parameters

Use the following configuration parameters to configure logical logs.

Table 3. Logical log configuration parameters

Configuration Parameter	Reference
LOGFILES configuration parameter on page 120	The number of logical log files.

Table 3. Logical log configuration parameters (continued)

Configuration Parameter	Reference
LOGSIZE configuration parameter on page 122	The size of each logical log file.
DYNAMIC_LOGS configuration parameter on page 95	The type of dynamic log allocation.
LOGBUFF configuration parameter on page 119	The size of the logical log buffer.

Long transaction configuration parameters

Use the following configuration parameters to control when long transactions are rolled back.

Table 4. Long transaction configuration parameters

Configuration Parameter	Reference
LTXHWM configuration parameter on page 126	The percentage of the logical log files that can be filled before a long transaction is rolled back.
LTXEHWM configuration parameter on page 125	The percentage of the logical log files that can be filled before the server suspends other activities so that a long transaction has exclusive use of the logs.

Server message file configuration parameters

Use the following configuration parameters to configure the server message file.

Table 5. Server message file configuration parameters

Configuration Parameter	Reference
MSGPATH configuration parameter on page 131	The path of the message file.
CONSOLE configuration parameter on page 64	The path of the console message file.

Tblspace configuration parameters

Use the following configuration parameters to configure the **tblspace** in the root dbspace.

Table 6. Tblspace configuration parameters

Configuration Parameter	Reference
TBLTBLFIRST configuration parameter on page 200	The first extent size for the tblspace **tblspace**.

Table 6. Tblspace configuration parameters (continued)

Configuration Parameter	Reference
TBLTBLNEXT configuration parameter on page 201	The next extent size for the tblspace **tblspace**.
TBLSPACE_STATS configuration parameter on page 200	Enables or disables tblspace statistics.

Temporary dbspace and sbspace configuration parameters

Use the following configuration parameters to configure the default temporary dbspaces and sbspaces.

Table 7. Temporary dbspace and sbspace configuration parameters

Configuration Parameter	Reference
DBSPACETEMP configuration parameter on page 70	The list of dbspaces for temporary objects.
SBSPACETEMP configuration parameter on page 160	The list of sbspaces for temporary smart large objects.

Dbspace and sbspace configuration parameters

Use the following configuration parameters to configure the default dbspaces and sbspaces.

Table 8. Default dbspaces and sbspaces configuration parameters

Configuration Parameter	Reference
SBSPACENAME configuration parameter on page 158	The default sbspace to store smart large objects.
SYSSBSPACENAME configuration parameter on page 198	The default sbspace for system statistics.
ONDBSPACEDOWN configuration parameter on page 138	Specifies the behavior of the server when a dbspace is down.

System configuration parameters

Use the following configuration parameters to set server instance information.

Table 9. System configuration parameters

Configuration Parameter	Reference
SERVERNUM configuration parameter on page 171	The unique ID for the database server instance.

Table 9. System configuration parameters (continued)

Configuration Parameter	Reference
DBSERVERNAME configuration parameter on page 69	The name of the default database server.
DBSERVERALIASES configuration parameter on page 68	List of alternative database server names.
FULL_DISK_INIT configuration parameter on page 106	Prevents an accidental disk reinitialization of an existing server instance.

Network configuration parameters

Use the following configuration parameters to configure the network.

Table 10. Network configuration parameters

Configuration Parameter	Reference
NETTYPE configuration parameter on page 133	The configuration of poll threads for a specific protocol.
LISTEN_TIMEOUT configuration parameter on page 117	The time the database server waits for a connection.
MAX_INCOMPLETE_CONNECTIONS configuration parameter on page 127	The maximum number of incomplete connections.
FASTPOLL configuration parameter on page 105	Enables or disables fast polling.
NUMFDSERVERS configuration parameter on page 136	For network connections on UNIX™, use the NUMFDSERVERS configuration parameter to specify the maximum number of poll threads to handle network connections that are moving between VPs.
NS_CACHE configuration parameter on page 135	Defines the maximum retention time for an individual entry in the host name/IP address cache, the service cache, the user cache, and the group cache.

CPU-related configuration parameters

Use the following configuration parameters to configure CPU virtual processors.

Table 11. CPU virtual processors configuration parameters

Configuration Parameter	Reference
MULTIPROCESSOR configuration parameter on page 132	Setting of `1` supports multiple CPU VPs.

Table 11. CPU virtual processors configuration parameters (continued)

Configuration Parameter	Reference
VPCLASS configuration parameter on page 212	Defines the properties of each CPU virtual processor class.
VP_MEMORY_CACHE_KB configuration parameter on page 210	The amount of private memory blocks for the CPU virtual processors.
SINGLE_CPU_VP configuration parameter on page 181	Set to `0` to enable user-defined CPU VPs, or `1` for a single CPU VP.

Automatic tuning configuration parameters

Use the following configuration parameters to automatically tune the configuration of the database server.

Table 12. CPU virtual processors configuration parameters

Configuration Parameter	Reference
AUTO_TUNE configuration parameter on page 45	Enable or disables all automatic tuning configuration parameters that have values that are not present in the `onconfig` file.
AUTO_LRU_TUNING configuration parameter on page 41	Enables or disables automatic tuning of LRU queues:
AUTO_AIOVPS configuration parameter on page 37	Enables or disables automatic management of AIO virtual processors.
AUTO_CKPTS configuration parameter on page 38	Enables or disables automatic checkpoints.
AUTO_REPREPARE configuration parameter on page 43	Enables or disables automatically reoptimizing stored procedures and repreparing prepared statements.
AUTO_STAT_MODE configuration parameter on page 45	Enables or disables the mode for selectively updating statistics for your system.
AUTO_READAHEAD configuration parameter on page 42	Changes the automatic read-ahead mode or disables or enables automatic read ahead for a query.

AIO and cleaner-related configuration parameters

Use the following configuration parameters to configure AIO virtual processors and buffer cleaners.

Table 13. AIO and buffer cleaner configuration parameters

Configuration Parameter	Reference
VPCLASS configuration parameter on page 212	Configures the AIO virtual processors.

Table 13. AIO and buffer cleaner configuration parameters (continued)

Configuration Parameter	Reference
CLEANERS configuration parameter on page 62	The number of page cleaner threads.
AUTO_AIOVPS configuration parameter on page 37	Enables or disables automatic management of AIO virtual processors.
DIRECT_IO configuration parameter (UNIX) on page 76	Specifies whether to use direct I/O.

Lock-related configuration parameters

Use the following configuration parameters to set locking behavior.

Table 14. Locking configuration parameters

Configuration Parameter	Reference
LOCKS configuration parameter on page 118	The initial number of locks at startup.
DEF_TABLE_LOCKMODE configuration parameter on page 74	The default table lock mode.

Shared memory configuration parameters

Use the following configuration parameters to configure shared memory.

Table 15. Shared memory configuration parameters

Configuration Parameter	Reference
RESIDENT configuration parameter on page 150	Controls whether shared memory is resident.
SHMBASE configuration parameter on page 176	The shared memory base address. Do not change this value.
SHMVIRTSIZE configuration parameter on page 179	The initial size, in KB, of the virtual segment of shared memory.
SHMADD configuration parameter on page 174	The size of virtual shared memory segments.
EXTSHMADD configuration parameter on page 103	The size of each virtual-extension shared memory segment for user-defined routines and DataBlade® routines that run in user-defined virtual processors.
SHMTOTAL configuration parameter on page 177	The maximum amount of shared memory for the database server.

Table 15. Shared memory configuration parameters (continued)

Configuration Parameter	Reference
SHMVIRT_ALLOCSEG configuration parameter on page 178	Controls when to add a memory segment.
SHMNOACCESS configuration parameter on page 176	Lists shared memory addresses that the server cannot access.

Checkpoint and system block configuration parameters

Use the following configuration parameters to configure checkpoints, recovery time objective, and system block time.

Table 16. Checkpoints, recovery time objective, and system block time configuration parameters

Configuration Parameter	Reference
CKPTINTVL configuration parameter on page 61	How often to check if a checkpoint is needed.
AUTO_CKPTS configuration parameter on page 38	Enables or disables automatic checkpoints.
RTO_SERVER_RESTART configuration parameter on page 156	The recovery time objective for a restart after a failure.
BLOCKTIMEOUT configuration parameter on page 50	The amount of time for a system block.

Conversion guard configuration parameters

Use the following configuration parameters to control information OneDB uses during an upgrade to a new version of the server.

Table 17. Conversion guard configuration parameters

Configuration Parameter	Reference
CONVERSION_GUARD configuration parameter on page 65	Specifies whether to stop or continue an upgrade if an error occurs during the upgrade.
RESTORE_POINT_DIR configuration parameter on page 152	Specifies the path name to an empty directory where restore point files are placed during a failed upgrade when the CONVERSION_GUARD configuration parameter is enabled.

Transaction-related configuration parameters

Use the following configuration parameters to control distributed transactions.

Table 18. Distributed transaction configuration parameters

Configuration Parameter	Reference
TXTIMEOUT configuration parameter on page 204	The distributed transaction timeout period.
DEADLOCK_TIMEOUT configuration parameter on page 73	The maximum amount of time to wait for a lock in a distributed transaction.

Backup and restore configuration parameters

Use the following configuration parameters to control backup and restore with the ON-Bar utility. Unless specified otherwise, these configuration parameters are documented in the *HCL OneDB™ Backup and Restore Guide*.

Table 19. ON-Bar configuration parameters

Configuration Parameter	Reference
BAR_ACT_LOG configuration parameter on page	The location of the ON-Bar activity log file.
BAR_DEBUG_LOG configuration parameter on page	The location of the ON-Bar debug log file.
BAR_DEBUG configuration parameter on page	The debug level for ON-Bar.
BAR_MAX_BACKUP configuration parameter on page	The number of backup threads used in a backup.
BAR_MAX_RESTORE configuration parameter on page	The number of restore threads used in a restore.
BAR_RETRY configuration parameter on page	The number of times to try a backup or restore again.
BAR_NB_XPORT_COUNT configuration parameter on page	The number of data buffers each backup process uses.
BAR_XFER_BUF_SIZE configuration parameter on page	The size of each data buffer.
RESTARTABLE_RESTORE configuration parameter on page	Enables ON-Bar to continue a backup after a failure.
BAR_PROGRESS_FREQ configuration parameter on page	How often progress messages are put in the activity log.
BAR_BSALIB_PATH configuration parameter on page	The path for the shared library for ON-Bar and the storage manager.

Table 19. ON-Bar configuration parameters (continued)

Configuration Parameter	Reference
BACKUP_FILTER configuration parameter on page	The path of a filter program to use during backups.
RESTORE_FILTER configuration parameter on page	The path of a filter program to use during restores.
BAR_PERFORMANCE configuration parameter on page	The type of ON-Bar performance statistics to report.
BAR_CKPTSEC_TIMEOUT configuration parameter on page	Time in seconds to wait for an archive checkpoint to complete in the secondary server.

Primary Storage Manager configuration parameters

Use the following configuration parameters to configure the HCL OneDB™ Primary Storage Manager.

Table 20. OneDB® Primary Storage Manager configuration parameters

Configuration Parameter	Reference
PSM_ACT_LOG configuration parameter on page	Specifies the location of the OneDB® Primary Storage Manager activity log if you do not want the log information included in the ON-Bar activity log.
PSM_DEBUG_LOG configuration parameter on page	Specifies the location of the OneDB® Primary Storage Manager debug log if you do not want the log information included in the ON-Bar debug log.
PSM_DEBUG configuration parameter on page	Specifies the amount of information that prints in the OneDB® Primary Storage Manager debug log if you want to use a debug level that is different from the one used by ON-Bar.
PSM_CATALOG_PATH configuration parameter on page	Specifies the full path to the directory that contains the OneDB® Primary Storage Manager catalog tables.
PSM_DBS_POOL configuration parameter on page	Specifies the name of the pool in which theOneDB® Primary Storage Manager places backup and restore dbspace data.
PSM_LOG_POOL configuration parameter on page	Specifies the name of the pool in which theOneDB® Primary Storage Manager places backup and restore log data.

Data dictionary cache configuration parameters

Use the following configuration parameters to configure the data dictionary caches.

Table 21. Data dictionary cache configuration parameters

Configuration Parameter	Reference
DD_HASHSIZE configuration parameter on page 73	The number of hash buckets in the data dictionary cache.
DD_HASHMAX configuration parameter on page 72	The maximum number of tables in each hash bucket.

Data distribution configuration parameters

Use the following configuration parameters to configure the data distribution pools.

Table 22. Data distribution configuration parameters

Configuration Parameter	Reference
DS_HASHSIZE configuration parameter on page 85	The number of hash buckets in the data distribution cache and other caches.
DS_POOLSIZE configuration parameter on page 89	The maximum number of entries in the data distribution cache and other caches.

User defined routine (UDR) configuration parameters

Use the following configuration parameters to configure UDRs.

Table 23. UDR configuration parameters

Configuration Parameter	Reference
PC_HASHSIZE configuration parameter on page 142	The number of hash buckets in the UDR cache.
PC_POOLSIZE configuration parameter on page 143	The maximum number of entries in the UDR cache.
PRELOAD_DLL_FILE configuration parameter on page 147	The C UDR shared library path name to load when the server starts.

SQL statement cache configuration parameters

Use the following configuration parameters to configure the SQL statement cache.

Table 24. SQL statement cache configuration parameters

Configuration Parameter	Reference
STMT_CACHE configuration parameter on page 192	Controls SQL statement caching.

Table 24. SQL statement cache configuration parameters (continued)

Configuration Parameter	Reference
STMT_CACHE_HITS configuration parameter on page 193	The number of times an SQL statement is run before it is cached.
STMT_CACHE_SIZE configuration parameter on page 195	The size of the SQL statement cache.
STMT_CACHE_NOLIMIT configuration parameter on page 194	Controls additional memory consumption of the SQL statement cache.
STMT_CACHE_NUMPOOL configuration parameter on page 194	The number of pools for the SQL statement cache.

Operating system session-related configuration parameters

Use the following configuration parameters to configure operating system and session features.

Table 25. Operating system and session configuration parameters

Configuration Parameter	Reference
USEOSTIME configuration parameter on page 207	The precision of SQL statement timing.
STACKSIZE configuration parameter on page 190	The size of a session stack.
ALLOW_NEWLINE configuration parameter on page 36	Whether embedded new line characters are allowed in SQL statements.
USELASTCOMMITTED configuration parameter on page 206	Controls committed read isolation level.

Index-related configuration parameters

Use the following configuration parameters to configure index features.

Table 26. Index configuration parameters

Configuration Parameter	Reference
FILLFACTOR configuration parameter on page 105	The percentage of index page fullness.
MAX_FILL_DATA_PAGES configuration parameter on page 127	Enables or disables filling data pages as full as possible if they have variable length rows.
BTSCANNER Configuration Parameter on page 50	Configures B-tree scanner threads.

Table 26. Index configuration parameters (continued)

Configuration Parameter	Reference
ONLIDX_MAXMEM configuration parameter on page 139	The amount of memory for the pre-image and updator log pools.

Parallel database queries (PDQ) configuration parameters

Use the following configuration parameters to configure PDQ.

Table 27. PDQ configuration parameters

Configuration Parameter	Reference
MAX_PDQPRIORITY configuration parameter on page 128	The maximum percentage of resources for a single query.
DS_MAX_QUERIES configuration parameter on page 86	The maximum number of concurrent decision support queries.
DS_TOTAL_MEMORY configuration parameter on page 90	The maximum amount of decision support memory.
DS_MAX_SCANS configuration parameter on page 87	The maximum number of decision support scans.
DS_NONPDQ_QUERY_MEM configuration parameter on page 88	The amount of non-PDQ query memory.
DATASKIP Configuration Parameter on page 66	Whether to skip a dbspace when processing a query.

Optimizer configuration parameters

Use the following configuration parameters to influence query execution optimizer plans and directives.

Table 28. Optimizer configuration parameters

Configuration Parameter	Reference
OPTCOMPIND configuration parameter on page 140	Controls how the optimizer determines the best query path.
DIRECTIVES configuration parameter on page 77	Enables or disables inline optimizer directives.
EXT_DIRECTIVES configuration parameter on page 102	Enables or disables external directives.
OPT_GOAL configuration parameter on page 141	Controls how to optimize for fastest retrieval.

Table 28. Optimizer configuration parameters (continued)

Configuration Parameter	Reference
IFX_FOLDVIEW configuration parameter on page 112	Enables or disables folding views.
AUTO_REPREPARE configuration parameter on page 43	Enables or disables automatically reoptimizing stored procedures and repreparing prepared statements.
AUTO_STAT_MODE configuration parameter on page 45	Enables or disables the mode for selectively updating statistics for your system.
STATCHANGE configuration parameter on page 192	Specifies a positive integer for a global percentage of a change threshold to identify data distribution statistics that need to be updated.
USTLOW_SAMPLE configuration parameter on page 210	Enables or disables the generation of index statistics based on sampling when you run UPDATE STATISTICS statements in LOW mode.

Scan configuration parameters

Use the following configuration parameters to set read-ahead behavior.

Table 29. Scan configuration parameters

Configuration Parameter	Reference
BATCHEDREAD_TABLE configuration parameter on page 49	Enables or disables light scans on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data.
BATCHEDREAD_INDEX configuration parameter on page 48	Enables the optimizer to perform light scans for indexes.
AUTO_READAHEAD configuration parameter on page 42	Changes the automatic read-ahead mode or disables or enables automatic read ahead for a query.

SQL tracing and EXPLAIN plan configuration parameters

Use the following configuration parameters to set SQL tracing.

Table 30. SQL tracing and EXPLAIN plan configuration parameters

Configuration Parameter	Reference
EXPLAIN_STAT configuration parameter on page 102	Enables or disables including query statistics in the explain output file.
SQLTRACE configuration parameter on page 189	Configures SQL tracing.

Security configuration parameters

Use the following configuration parameters to configure security options.

Table 31. Security configuration parameters

Configuration Parameter	Reference
DBCREATE_PERMISSION configuration parameter on page 67	Specifies users who can create databases.
IFX_EXTEND_ROLE configuration parameter on page 111	Controls how to specify which users can register external routines.
SECURITY_LOCALCONNECTION configuration parameter on page 170	Whether the database server checks the security of local connections.
UNSECURE_ONSTAT configuration parameter on page 204	Whether non-DBSA users can run **onstat** commands.
ADMIN_USER_MODE_WITH_DBSA configuration parameter on page 34	Controls who can connect to the server in administration mode.
ADMIN_MODE_USERS configuration parameter on page 33	Lists the users who can connect in administration mode.
DISK_ENCRYPTION configuration parameter on page 79	Controls the encryption of storage spaces.

Label-based access control configuration parameters

Use the following configuration parameters to configure the label-based access control (LBAC) cache. These configuration parameters are documented in the *HCL OneDB™ Security Guide*.

Table 32. LBAC configuration parameters

Configuration Parameter	Reference
PLCY_POOLSIZE configuration parameter on page 145	The number of hash buckets in the LBAC security information cache.
PLCY_HASHSIZE configuration parameter on page 145	The maximum number of entries in each hash bucket of the LBAC security information cache.
USRC_POOLSIZE configuration parameter on page 209	The number of hash buckets in the LBAC credential memory cache.
USRC_HASHSIZE configuration parameter on page 209	The maximum number of entries in each hash bucket of the LBAC credential memory cache.

Optical configuration parameters

Use the following configuration parameters to configure the optical storage subsystem.

Table 33. Optical storage subsystem configuration parameters

Configuration Parameter	Reference
STAGEBLOB configuration parameter on page 191	The name of the optical blobspace.
OPCACHEMAX configuration parameter (UNIX) on page 140	The maximum size of the optical cache.

Built-in character data types configuration parameters

Use the following configuration parameter to configure built-in character data types.

Table 34. Built-in character data types configuration parameters

Configuration Parameter	Reference
SQL_LOGICAL_CHAR configuration parameter on page 187	Enables or disables the expansion of size specifications in declarations of built-in character data types.

Sequence cache configuration parameters

Use the following configuration parameter to configure the sequence cache:

Table 35. Sequence cache data types configuration parameters

Configuration Parameter	Reference
SEQ_CACHE_SIZE configuration parameter on page 170	Specifies the maximum number of sequence objects that are cached in memory.

High-availability and Enterprise Replication security configuration parameters

Use the following configuration parameters to configure security for high-availability clusters and Enterprise Replication.

Table 36. High-availability and Enterprise Replication security configuration parameters

Configuration Parameter	Reference
ENCRYPT_HDR configuration parameter on page 99	Enables or disables encryption for HDR.
ENCRYPT_SMX configuration parameter on page 101	The level of encryption for SDS or RSS servers.
ENCRYPT_CDR Configuration Parameter on page	The level of encryption for Enterprise Replication.

Table 36. High-availability and Enterprise Replication security configuration parameters (continued)

Configuration Parameter	Reference
ENCRYPT_CIPHERS configuration parameter on page 97	Lists encryption ciphers and modes.
ENCRYPT_MAC configuration parameter on page 99	The level of the message authentication code (MAC).
ENCRYPT_MACFILE configuration parameter on page 100	The paths of MAC key files.
ENCRYPT_SWITCH configuration parameter on page 101	The frequency to switch ciphers and keys.

Enterprise Replication configuration parameters

Use the following configuration parameters to configure Enterprise Replication (ER). These configuration parameters are documented in the *HCL OneDB™ Enterprise Replication Guide*.

Table 37. Enterprise Replication configuration parameters

Configuration Parameter	Reference
CDR_EVALTHREADS Configuration Parameter on page	The numbers of evaluator threads.
CDR_DSLOCKWAIT Configuration Parameter on page	The amount of time data sync threads wait for database locks.
CDR_QUEUEMEM Configuration Parameter on page	The maximum amount of memory for send and receive queues.
CDR_NIFCOMPRESS Configuration Parameter on page	The network interface compression level.
CDR_SERIAL Configuration Parameter on page	The incremental size and starting value of serial columns.
CDR_DBSPACE Configuration Parameter on page	The dbspace name for the **syscdr** database.
CDR_QDATA_SBSPACE Configuration Parameter on page	The names of sbspaces for spooled transactions.
CDR_SUPPRESS_ATSRISWARN Configuration Parameter on page	The data sync warnings and errors to suppress in ATS and RIS files.
CDR_DELAY_PURGE_DTC configuration parameter on page	The amount of time to retain delete tables.

Table 37. Enterprise Replication configuration parameters (continued)

Configuration Parameter	Reference
CDR_LOG_LAG_ACTION configuration parameter on page	The action taken when the database server comes close to overwriting a logical log that Enterprise Replication did not yet process.
CDR_LOG_STAGING_MAXSIZE Configuration Parameter on page	The maximum amount of space that Enterprise Replication uses to stage log files.
CDR_MAX_DYNAMIC_LOGS Configuration Parameter on page	The maximum number of dynamic log requests that Enterprise Replication can make in a session.
GRIDCOPY_DIR Configuration Parameter on page	The default directory used by the ifx_grid_copy procedure.
CDR_TSINSTANCEID configuration parameter on page	The unique identifier for time series instances that are replicated.
CDR_MAX_FLUSH_SIZE configuration parameter on page	The maximum number of transactions that are applied before the logs are flushed to disk.
CDR_AUTO_DISCOVER configuration parameter on page	Allow auto-configuration of Enterprise Replication though the cdr autoconfig serv command, installation wizard, or ifxclone utility.
CDR_MEM configuration parameter on page	Specifies the method of memory pool allocation for Enterprise Replication.

Parallel sharded queries configuration parameters

Use the following configuration parameters to configure parallel sharded queries.

Table 38. Parallel sharded queries configuration parameters

Configuration Parameter	Reference
SHARD_MEM configuration parameter on page	Specifies how to allocate shared memory for sharded queries on a shard server.
SHARD_ID configuration parameter on page	Sets the unique ID for a shard server in a shard cluster.

High-availability cluster configuration parameters

Use the following configuration parameters to configure high-availability clusters.

Table 39. High-availability cluster configuration parameters

Configuration Parameter	Reference
DRAUTO configuration parameter on page 81	Controls automatic failover of primary servers.

Table 39. High-availability cluster configuration parameters (continued)

Configuration Parameter	Reference
DRINTERVAL configuration parameter on page 82	The maximum interval between buffer flushes.
HDR_TXN_SCOPE configuration parameter on page 110	Adjust transaction synchronization between client applications, the primary server, and the HDR secondary server.
DRTIMEOUT configuration parameter on page 84	The network timeout period.
DRLOSTFOUND configuration parameter on page 84	The path of the HDR lost-and-found file.
DRIDXAUTO configuration parameter on page 82	Enables or disables automatic index repair.
HA_ALIAS configuration parameter on page 106	The server alias for a high-availability cluster.
HA_FOC_ORDER configuration parameter on page 107	Defines a single failover rule used by Connection Managers.
LOG_INDEX_BUILDS configuration parameter on page 121	Enables or disables index page logging.
SDS_ENABLE configuration parameter on page 161	Enables or disables and SD secondary server.
SDS_TIMEOUT configuration parameter on page 166	The time the primary waits for acknowledgment from an SD secondary server.
SDS_TEMPDBS configuration parameter on page 165	The temporary dbspace used by an SD secondary server.
SDS_ALTERNATE configuration parameter on page 160	The alternate means of communication between the primary server and SD secondary servers in a high-availability cluster.
SDS_PAGING configuration parameter on page 164	The paths of SD secondary paging files.
SDS_LOGCHECK configuration parameter on page 163	Whether the primary server is generating log activity and to allow or prevent failover of the primary server.
UPDATABLE_SECONDARY configuration parameter on page 205	Whether the secondary server can accept update, insert, or delete operations from clients.
FAILOVER_CALLBACK configuration parameter on page 103	The program called when a secondary server makes the transition to a standard or primary server.

Table 39. High-availability cluster configuration parameters (continued)

Configuration Parameter	Reference
TEMPTAB_NOLOG configuration parameter on page 201	The default logging mode for temporary tables.
DELAY_APPLY Configuration Parameter on page 75	The delay time for applying transactions on an RS secondary server.
STOP_APPLY configuration parameter on page 195	Stops applying transactions on an RS secondary server.
LOG_STAGING_DIR configuration parameter on page 121	The directory to stage log files.
RSS_FLOW_CONTROL configuration parameter on page 155	Enables flow control for RS secondary servers.
SDS_FLOW_CONTROL configuration parameter on page 162	Enables flow control for SD secondary servers.
FAILOVER_TX_TIMEOUT configuration parameter on page 104	Enables or disables transaction survival behavior during failover.
ENABLE_SNAPSHOT_COPY configuration parameter on page 97	Whether the server instance can be cloned by the ifxclone utility.
SMX_COMPRESS configuration parameter on page 182	The level of compression that the database server uses when sending data from the source database server to the target database server.
SMX_PING_INTERVAL configuration parameter on page 183	The number of seconds in a timeout interval.
SMX_PING_RETRY configuration parameter on page 184	The number of timeout intervals before a secondary server closes the SMX connection to the primary server.
CLUSTER_TXN_SCOPE configuration parameter on page 63	Controls when transaction commits can be returned to a client application.
SMX_NUMPIPES configuration parameter on page 182	Sets the number of pipes for SMX connections.
SEC_NONBLOCKING_CKPT configuration parameter on page 169	Enables non-blocking checkpoint at HDR and RS secondary server.

Logical recovery configuration parameters

Use the following configuration parameters to set logical recovery threads.

Table 40. Logical recovery configuration parameters

Configuration Parameter	Reference
ON_RECVRY_THREADS configuration parameter on page 138	The number of logical recovery threads that run in parallel during a warm restore.
OFF_RECVRY_THREADS configuration parameter on page 137	The number of logical recovery threads used in a cold restore and for fast recovery.

Diagnostic dump configuration parameters

Use the following configuration parameters to control diagnostic dump information.

Table 41. Diagnostic configuration parameters

Configuration Parameter	Reference
DUMPDIR configuration parameter on page 93	The location of assertion failure diagnostic files.
DUMPSHMEM configuration parameter (UNIX) on page 94	Controls shared memory dumps.
DUMPGCORE configuration parameter (UNIX) on page 93	Enables or disables whether the database server dumps a core to the **gcore** file.
DUMPCORE configuration parameter (UNIX) on page 92	Enables or disables whether the database server dumps a core after an assertion failure.
DUMPCNT configuration parameter (UNIX) on page 91	The maximum number of shared memory dumps for a session.

Alarm program configuration parameters

Use the following configuration parameters to configure the alarm program.

Table 42. Alarm program configuration parameters

Configuration Parameter	Reference
ALARMPROGRAM configuration parameter on page 34	The alarm program to display event alarms.
ALRM_ALL_EVENTS configuration parameter on page 36	Whether the alarm program runs for all events.
STORAGE_FULL_ALARM configuration parameter on page 196	How often messages and events are raised when a storage space is full or a partition runs out of pages or extents.
SYSALARMPROGRAM configuration parameter on page 197	The system alarm program triggered after an assertion failure.

Technical support configuration parameters

The following configuration parameters to are used by technical support and are set automatically.

Table 43. Technical support configuration parameters

Configuration Parameter	Reference
RAS_PLOG_SPEED	Reserved for support.
RAS_LLOG_SPEED	Reserved for support.

Character processing configuration parameter

Use the following configuration parameter to control whether HCL OneDB™ checks if characters are valid for the locale.

Table 44. Character processing configuration parameter

Configuration Parameter	Reference
EILSEQ_COMPAT_MODE configuration parameter on page 96	Enables or disables checking character validity.

Statistics configuration parameters

Use the following configuration parameters to control the collection of queue and wait statistics.

Table 45. Queue and wait statistics configuration parameters

Configuration Parameter	Reference
QSTATS configuration parameter on page 148	Enables or disables collecting queue statistics.
WSTATS configuration parameter on page 216	Enables or disables collecting wait statistics.

User mapping configuration parameter

Use this configuration parameter to control user mapping.

Table 46. User mapping

Configuration Parameter	Description
USERMAPPING configuration parameter (UNIX, Linux) on page 208	Whether mapped users can connect to HCL OneDB™, and if so, whether the mapped user can have administrative privileges.

Storage provisioning configuration parameters

Use the following configuration parameters to control information that enables the server to automatically extend or add a chunk when more space is needed in an existing storage space (dbspace, temporary dbspace, sbspace, temporary sbspace, or blobspace).

Table 47. Storage provisioning configuration parameters

Configuration Parameter	Reference
SP_AUTOEXPAND configuration parameter on page 185	Enables or disables the automatic creation or extension of chunks in a storage space.
SP_THRESHOLD configuration parameter on page 185	Defines the minimum amount of free KB that can exist in a storage space.
SP_WAITTIME configuration parameter on page 186	Specifies the maximum number of seconds that a thread waits for a storage pool to expand before returning an "out of space" error.

Automatic location of database objects

Use the following configuration parameter to enable automatic location and fragmentation.

Table 48. Automatic location configuration parameter

Configuration Parameter	Reference
AUTOLOCATE configuration parameter on page 47	Enables the automatic location of databases and tables and the automatic fragmentation of tables.

Default escape character for LIKE/MATCHESconfiguration parameter

Use the following configuration parameter as needed.

Table 49. Default escape configuration parameter

Configuration Parameter	Reference
DEFAULTESCCHAR configuration parameter on page 75	Specifies a default escape character.

Non-root user server installation configuration parameters

Use the following configuration parameters with non-root server installations.

Table 50. Non-root user server installation

Configuration Parameter	Reference
REMOTE_SERVER_CFG configuration parameter on page 148	Specifies the name of a file that lists the remote hosts that are trusted by the database server computer.

Table 50. Non-root user server installation (continued)

Configuration Parameter	Reference
REMOTE_USERS_CFG configuration parameter on page 149	Specifies the name of a file that lists names of trusted users that exist on remote hosts.
S6_USE_REMOTE_SERVER_CFG configuration parameter on page 157	Specifies the file used to authenticate secure server connections in a trusted network environment.

Low memory configuration parameters

Use the following configuration parameters to manage low memory.

Table 51. Low memory configuration parameters

Configuration Parameter	Reference
LOW_MEMORY_RESERVE configuration parameter on page 124	Reserves a specific amount of memory for use when critical activities are needed and the server has limited free memory.
LOW_MEMORY_MGR configuration parameter on page 123	Change the default behavior of the server when it reaches the memory limit.

Connection parameters

Use the following parameters to manage connections.

Table 52. Connection configuration parameters

Configuration Parameter	Description
CONNECT_RETRIES configuration parameter on page 113	Specifies the number of connection attempts that can be made to the database server after the initial connection attempt fails. With the CONNECT_TIMEOUT configuration parameter, specifies the frequency at which the CONNECT statement tries to connect to the database server.
CONNECT_TIMEOUT configuration parameter on page 114	Specifies the duration, in seconds, that the CONNECT statement attempts to establish a connection to the database server. With the INFORMIXRETRY configuration parameter, specifies the frequency at which the CONNECT statement tries to connect to the database server.

Session limits

Use the following configuration parameters to create limits for individual sessions.

Table 53. Session-limit configuration parameters.

Configuration Parameter	Reference
SESSION_LIMIT_LOCKS configuration parameter on page 171	Limits the number of locks.
SESSION_LIMIT_MEMORY configuration parameter on page 172	Limits the available memory.
SESSION_LIMIT_TEMPSPACE configuration parameter on page 173	Limits temporary table space.
SESSION_LIMIT_LOGSPACE configuration parameter on page 172	Limits logspace available to individual transactions.
SESSION_LIMIT_TXN_TIME configuration parameter on page 174	Limits the amount of time that a transaction can run.

Tenant limits

Use the following configuration parameters to specify limits on tenant databases.

Table 54. Tenant limits configuration parameters.

Configuration Parameter	Reference
TENANT_LIMIT_SPACE configuration parameter on page 203	Limits the amount of storage space available to a tenant database.
TENANT_LIMIT_MEMORY configuration parameter on page 203	Limits the amount of shared memory for all sessions that are connected to the tenant database.
TENANT_LIMIT_CONNECTIONS configuration parameter on page 202	Limits the number of connections to a tenant database.

Java™ configuration parameters

Use the following configuration parameters to configure Java™ virtual processors. These configuration parameters are documented in the *HCL® J/Foundation Developer's Guide*.

Table 55. Java™ configuration parameters

Configuration Parameter	Reference
VPCLASS	Configures a Java™ virtual processor class.
JVPPROPFILE	The Java™ VP property file.
JVPLOGFILE	The Java™ VP log file.
JVPARGS	Configures the Java™ VM.

Table 55. Java™ configuration parameters (continued)

Configuration Parameter	Reference
JVPCLASSPATH	The Java™ class path.

Buffer pool and LRU tuning configuration parameters

Use the following configuration parameters to configure buffer pools and tune LRU queues.

Table 56. Buffer pool and LRU tuning configuration parameters

Configuration Parameter	Reference
BUFFERPOOL configuration parameter on page 51	Configures buffer pools.
AUTO_LRU_TUNING configuration parameter on page 41	Enables or disables automatic tuning of LRU queues.

Additional parameters

Some configuration parameters are not in the `onconfig.std` file. You can add these parameters to your `onconfig` file as necessary.

Table 57. Parameters that are not in the onconfig.std file

Configuration Parameter	Reference
AUTO_TUNE_SERVER_SIZE configuration parameter on page 40	Sets the size of the database server based on the number of expected users. If you create a server during installation, this parameter is set in your `onconfig` file.
AUTO_LLOG configuration parameter on page 38	Automatically adds logical logs in the specified dbspace to improve performance and to limit the total size of logical log files. If you create a server during installation, this parameter is set in your `onconfig` file.
CDR_APPLY Configuration Parameter on page	Specifies the minimum and maximum number of data sync threads.
CDR_ENV Configuration Parameter on page	Sets some specific Enterprise Replication environment variables.
CHECKALLDOMAINSFORUSER configuration parameter on page 61	Specifies how the database server searches for user names in a networked Windows™ environment.
DISABLE_B162428_XA_FIX configuration parameter on page 78	Specifies whether to free global transactions after a rollback operation.

Table 57. Parameters that are not in the onconfig.std file (continued)

Configuration Parameter	Reference
DRDA_COMMBUFFSIZE configuration parameter on page 80	Specifies the size of the DRDA® communications buffer.
IFX_XA_UNIQUEXID_IN_DATABASE configuration parameter on page 112	Enables the transaction manager to use same XID to represent global transactions on different databases in the same database server instance.
LIMITNUMSESSIONS configuration parameter on page 116	Specifies the maximum number of sessions that can connect to the database server.
MSG_DATE configuration parameter on page 131	Inserts a date stamp at the beginning of messages that are printed to the online log.
NET_IO_TIMEOUT_ALARM configuration parameter on page 132	Sends notification if network write operations are blocked for 30 minutes or more.
PN_STAGEBLOB_THRESHOLD configuration parameter on page 146	Reserves space for BYTE and TEXT data in round-robin fragments.

ADMIN_MODE_USERS configuration parameter

The ADMIN_MODE_USERS configuration parameter specifies a list of users, besides the user **informix** and members of the DBSA group, that you want to access the database server in the administration mode.

onconfig.std value

Not set. Only user informix and members of the DBSA group can access HCL OneDB™ in administration mode.

separators

Comma-separated user names, such as: `Karin,Sarah,Andrew`, as a string of up to 127 bytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The list of users is in the ADMIN_MODE_USERS configuration parameter is preserved indefinitely. You can use the onmode -wm or onmode -wf command to remove users.

Use the onmode -j -U command to allow one or more users to access the database server in administration mode when the database is running.

You must set the ADMIN_USER_MODE_WITH_DBSA configuration parameter to 1 to enable the users that are listed in the ADMIN_MODE_USERS configuration parameter to connect to the database server in the administration mode.

ADMIN_USER_MODE_WITH_DBSA configuration parameter

The ADMIN_USER_MODE_WITH_DBSA configuration parameter specifies which users, besides the user **informix**, can connect to the database server in the administration mode.

onconfig.std value

Not set. Only the user **informix** can connect to the database server in administration mode.

values

`0` = Only the user **informix** can connect in the administration mode

`1` = If the ADMIN_USER_MODE configuration parameter is not set, the following users can connect in the administration mode:

- The user **informix**
- Members of the DBSA group

If the ADMIN_USER_MODE configuration parameter is set to a list of one or more user names, then following users can connect in the administration mode:

- The user **informix**
- The users who have the **informix** group included in their group list (UNIX™ only)
- Members of the DBSA group
- The administration users that are listed in the ADMIN_MODE_USERS configuration parameter

takes effect

After you edit your `onconfig` file and restart the database server.

ALARMPROGRAM configuration parameter

Use the ALARMPROGRAM configuration parameter to specify the full pathname of the `alarmprogram` file that handles event alarms and controls logical-log backups.

onconfig.std value

On UNIX™: `$ONEDB_HOME/etc/alarmprogram.sh`

On Windows™: `%ONEDB_HOME%\etc\alarmprogram.bat`

if not present

On UNIX™: `$ONEDB_HOME/etc/no_log.sh`

On Windows™: `%ONEDB_HOME%\etc\no_log.bat`

value

pathname = Full path name of the `alarmprogram` file.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

You can set the ALRM_ALL_EVENTS configuration parameter to specify whether the ALARMPROGRAM configuration parameter runs for all events that are logged in the MSGPATH, or only for specified noteworthy events (events greater than severity 1).

If the script that the ALARMPROGRAM configuration parameter specifies does not exist, the default alarm handler, `no_log.sh` or `no_log.bat`, is substituted. After you have the correct script in place, update the value of the ALARMPROGRAM configuration parameter to specify the script. You can make this update with the server online by using the onmode -wm command.

The following sample scripts are provided.

Table 58. Sample scripts

Script name (UNIX™)	Script name (Windows™)	Description
`log_full.sh`	`log_full.bat`	To back up logical logs automatically when the database server issues a log-full event alarm, set ALARMPROGRAM to `log_full.sh` or `log_full.bat`. You can modify the script and set it to the full path of ALARMPROGRAM in the `onconfig` file.
`no_log.sh`	`no_log.bat`	To disable automatic logical-log backups, set ALARMPROGRAM to `no_log.sh` or `no_log.bat`.
`alarmprogram.sh`	`alarmprogram.bat`	Handles event alarms and controls logical-log backups. Modify `alarmprogram.sh` or `alarmprogram.bat` and set ALARMPROGRAM to the full path name of `alarmprogram.sh` or `alarmprogram.bat`. See Customizing the ALARMPROGRAM Scripts.

Instead of using the supplied scripts, you can write your own shell script, batch file, or binary program to execute events. Set ALARMPROGRAM to the full pathname of this file. The database server executes this script when noteworthy events occur. These events include database, table, index, or simple-large-object failure; all logs are full; internal subsystem failure; initialization failure; and long transactions. You can have the events noted in an email or pagermail message.

To generate event alarms, set ALARMPROGRAM to `$ONEDB_HOME/etc/alarmprogram.sh` or `%ONEDB_HOME%\etc\alarmprogram.bat` and modify the file according.

Important: When you choose automatic logical-log backups, backup media should always be available for the backup process.

Do not use the continuous log backup command (onbar -b -l -C) if you have automatic log backup setup through the ALARMPROGRAM parameter.

ALLOW_NEWLINE configuration parameter

Use the ALLOW_NEWLINE configuration parameter to allow or disallow newline characters in quoted strings for all sessions.

To allow all remote sessions in a distributed query to support embedded newline characters, specify ALLOW_NEWLINE in their `onconfig` files.

onconfig.std value

ALLOW_NEWLINE 0

values

`0` = Disallow the newline character in quoted strings for all sessions.

`1` = Allow the newline character in quoted strings for all sessions.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

You can specify that you want the database server to allow the newline character (**\n**) in a quoted string either for all sessions or for a specific session. A session is the duration of a client connection to the database server.

To allow or disallow newline characters in quoted strings for the current session when ALLOW_NEWLINE is not set, you can execute the built-in ifx_allow_newline() routine with `'t'` or `'f'` as its only argument.

- `'t'` enables support for newline characters within quoted strings.
- `'f'` has the opposite effect.

Calls to ifx_allow_newline() affect only the user session from which that routine is invoked.

ALRM_ALL_EVENTS configuration parameter

Use the ALRM_ALL_EVENTS configuration parameter to specify whether the ALARMPROGRAM configuration parameter runs for all events that are logged in the MSGPATH configuration parameter, or only for noteworthy events.

onconfig.std value

ALRM_ALL_EVENTS 0

values

`0` = Only for noteworthy events.

`1` = The parameter triggers the ALARMPROGRAM configuration parameter and the ALRM_ALL_EVENTS configuration parameter displays all event alarms.

takes effect

After you edit your `onconfig` file and restart the database server.

AUTO_AIOVPS configuration parameter

The AUTO_AIOVPS configuration parameter enables the database server to automatically increase the number of asynchronous I/O virtual processors (AIO VPs) and page cleaner threads when the database server detects that the I/O workload outpaced the performance of the existing AIO VPs.

onconfig.std value

AUTO_AIOVPS 1

Not set. If the AUTO_TUNE configuration parameter is set to 1, AIO VPs and page cleaner threads are automatically increased.

values

`0` = Off

`1` = On

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

If an AUTO_AIOVPS value is not set in your current `onconfig` file and you edit the AUTO_TUNE configuration parameter and restart the database server

Usage

The VPCLASS **aio** configuration parameter controls the number of AIO VPs, If the VP **aio** parameter is not set in the `onconfig` file, the initial number of AIO VPs the database server starts when AUTO_AIOVPS is enabled is equal to the number of AIO chunks. The maximum number of AIO VPs the database server can start if VP **aio** is not set is 128.

AUTO_CKPTS configuration parameter

The AUTO_CKPTS configuration parameter allows the server to trigger checkpoints more frequently to avoid the blocking of transactions.

onconfig.std value

AUTO_CKPTS 1

Not set. If the AUTO_TUNE configuration parameter is set to 1, automatic checkpoints are enabled.

values

`0` = Off

`1` = On

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

If an AUTO_CKPTS value is not set in your current `onconfig` file and you edit the AUTO_TUNE configuration parameter and restart the database server

AUTO_LLOG configuration parameter

Use the AUTO_LLOG configuration parameter to automatically add logical logs in the specified dbspace to improve performance.

onconfig.std value

Not in the `onconfig.std` file.

default value if you created a server during installation

```
AUTO_LLOG 1,llog,max_size
```

The *max_size* value depends on the value of the AUTO_TUNE_SERVER_SIZE configuration parameter.

values

0 = Default. Disabled. Logical logs are not automatically added to improve performance.

1,*dbspace_name*,*max_size*

- 1 = Enabled. Logical logs are automatically added when needed to improve performance.
- *dbspace_name* = The name of the dbspace in which to add logical log files. The dbspace must have the default page size for the operating system.
- *max_size* = Optional. Default is 2048000 KB (2 GB). The maximum size, in KB, of all logical log files, including any logical log files that are not stored in the dbspace *dbspace_name*. When the maximum

size is reached, the database server no longer adds logical log files to improve performance. If *max_size* is not specified, the AUTO_TUNE_SERVER_SIZE configuration parameter setting affects the maximum size. See the Usage section.

separators

Separate fields with a comma.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If you created a server during installation, the AUTO_LLOG configuration parameter is enabled automatically. A dbspace that is named **llog** is created for logical logs. The installation program sets the initial size and value of the *max_size* option of the dbspace based on the value of the AUTO_TUNE_SERVER_SIZE configuration parameter. You can change the *max_size* option by resetting the value of the AUTO_LLOG configuration parameter.

If you did not create a server during installation, you can enable the AUTO_LLOG configuration parameter to automatically add logical log files when the database server detects that adding logical log files improves performance. For optimal performance, choose a dbspace on a separate disk from the root dbspace and the physical log.

When the AUTO_LLOG configuration parameter is enabled, the database server adds logical logs when the lack of logical logs causes too high a percentage of checkpoints, blocking checkpoints, or long checkpoints.

When the maximum size of the logical log files is reached, logical log files are no longer added to improve performance. However, if the DYNAMIC_LOGS configuration parameter is enabled, logical logs are added to prevent transaction blocking. The settings of the DYNAMIC_LOGS and the AUTO_LLOG configuration parameters do not interact. Similarly, you can continue to manually add logical log files.

If the value of the *max_size* field is larger than the size of the specified dbspace, make sure that your storage pool has available space.

Example

Example

The following setting enables the automatic addition of logical log files until size of all logical log files is 204800 KB and sets the dbspace for logical log files to **llog**:

```
AUTO_LLOG 1,llog,204800
```

AUTO_TUNE_SERVER_SIZE configuration parameter

Use the AUTO_TUNE_SERVER_SIZE configuration parameter to set the sizes of memory and storage spaces to allocate based on the number of expected concurrent users.

onconfig.std value

Not in the `onconfig.std` file.

Default value

Not set.

value if you created a server during installation

Depends on the number of users you specify in the installation program.

values

SMALL = 1 - 100 users

MEDIUM = 101 - 500 users

LARGE = 501 - 1000 users

XLARGE = more than 1000 users

takes effect

If you create a server during installation.

After you edit your `onconfig` file and restart the database server for the first time.

Usage

If you create a server during installation, you specify the number of expected users for the database server. The AUTO_TUNE_SERVER_SIZE configuration parameter is set to the corresponding size, which affects the size of the following properties:

- The size of the buffer pool.
- The maximum size of logical log files before the server stops automatically adding logical logs to improve performance
- The initial size of the following created storage spaces, which are created automatically during installation:
    - An extendable plogspace for the physical log
    - A dbspace for the logical log
    - Dbspaces for databases and tables
    - A temporary dbspace
    - An sbspace
    - A temporary sbspace

The following table shows how the value of the AUTO_TUNE_SERVER_SIZE configuration parameter affects sizes.

Table 59. Effect on memory and storage space allocations

Value	Maximum size of buffer pools (BUFFERPOOL)	Initial size of automatically created storage spaces	Maximum size of logical log files (AUTO_LLOG)
SMALL	10% of available shared memory	50 MB	200 MB
MEDIUM	20%	100 MB	500 MB
LARGE	33%	200 MB	1 GB
XLARGE	50%	500 MB	2 GB

If you did not create a server during installation, or you change the value of the AUTO_TUNE_SERVER_SIZE configuration parameter after you initialize the server for the first time, the new value affects the size of only the following properties:

- The size of the buffer pool, if the BUFFERPOOL configuration parameter setting includes the memory='auto' option.
- The maximum size of all logical log files before the server stops automatically adding logical logs to improve performance.

AUTO_LRU_TUNING configuration parameter

Use the AUTO_LRU_TUNING configuration parameter to enable automatic LRU tuning, which automatically maintains enough clean pages for page replacement.

onconfig.std value

AUTO_LRU_TUNING 1

Not set. If the AUTO_TUNE configuration parameter is set to 1, automatic LRU tuning is enabled.

values

`0` = Off

`1` = On

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

If an AUTO_LRU_TUNING value is not set in your current `onconfig` file and you edit the AUTO_TUNE configuration parameter and restart the database server

Usage

Automatic LRU tuning changes affect all buffer pools and adjust the **lru_min_dirty** and **lru_max_dirty** values in the BUFFERPOOL configuration parameter.

AUTO_READAHEAD configuration parameter

Use the AUTO_READAHEAD configuration parameter to change the automatic read-ahead mode or to disable automatic read-ahead operations for a query.

onconfig.std value

AUTO_READAHEAD 1

Not set. If the AUTO_TUNE configuration parameter is set to 1, read ahead is performed automatically in the standard mode.

values

An integer from 0 - 2 that specifies the mode, optionally followed by a comma and an integer that specifies the number of pages that are automatically requested to be read ahead. For example, the value `1,4096` enables automatic read-ahead in standard mode for 4096 pages at a time.

0 = Disable automatic read-ahead requests.

1 = Enable automatic read-ahead requests in the standard mode. The database server automatically processes read-ahead requests only when a query waits on I/O.

2 = Enable automatic read-ahead requests in the aggressive mode. The database server automatically processes read-ahead requests at the start of the query and continuously through the duration of the query.

number_of_pages = 4 - 4096, indicating the number of pages that are automatically requested to be read ahead. The default is 128 pages.

separators

Separate the mode and the number of pages with a comma.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

If an AUTO_READAHEAD value is not set in your current `onconfig` file and you edit the AUTO_TUNE configuration parameter and restart the database server

Usage

Automatic read-ahead operations help improve query performance by issuing asynchronous page requests when the database server detects that the query is encountering I/O. Asynchronous page requests can improve query performance by overlapping query processing with the processing necessary to retrieve data from disk and put it in the buffer pool.

Generally, the default value of `1` is appropriate for most production environments.

While there are no specific circumstances in which aggressive read-ahead operations perform significantly better than standard read-ahead operations, aggressive read-ahead might be slightly more effective:

- For some scans that read a small amount of data
- In situations in which you switch between turning read-ahead off for small scans and on for longer scans
- For scans that look only at a small number of rows, because the server performs read-ahead operations immediately rather than waiting for the scan to encounter I/O.

For scans that might turn read-ahead operations off and on because the scan hits pockets of cached data, aggressive read-ahead operations do not turn off read-ahead operations.

Use aggressive read-ahead operations only in situations in which you tested both settings and know that aggressive read-ahead operations are more effective. Do not use aggressive read-ahead operations if you are not sure that they are more effective.

You can use the AUTO_READAHEAD environment option of the SET ENVIRONMENT statement of SQL to enable or disable the value of the AUTO_READAHEAD configuration parameter for a session.

The precedence of read-ahead setting is as follows:

1. A SET ENVIRONMENT AUTO_READAHEAD statement for a session.
2. The AUTO_READAHEAD configuration parameter value of `1` or `2`.
3. If the value for the AUTO_READAHEAD configuration parameter is not present in the `onconfig` file, the server performs read-ahead on 128 data pages (which equates to AUTO_READAHEAD mode set to 1), when the server completes a query.

AUTO_REPREPARE configuration parameter

The AUTO_REPREPARE configuration parameter controls whether the database server automatically reoptimizes SPL routines and reprepares prepared objects after the schema of a table that is referenced by the SPL routine or by the prepared object was changed.

onconfig.std value

AUTO_REPREPARE 1

Not set. If the AUTO_TUNE configuration parameter is set to 1, SPL routines are automatically reoptimized and prepared objects are automatically reprepared.

values

`0` = Disables the automatic repreparation of prepared objects after the schema of a directly or an indirectly referenced table is modified. Also disables the automatic reoptimization of SPL routines after the schema of an indirectly referenced table is modified.

`1` = Enables automatic repreparation.

`3` = Enables automatic repreparation in optimistic mode.

`5` = Enables automatic repreparation on update statistics.

`7` = Enables automatic repreparation in optimistic mode and on update statistics.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

If an AUTO_REPREPARE value is not set in your current `onconfig` file and you edit the AUTO_TUNE configuration parameter and restart the database server

When you reset the value in memory by running the onmode -wm command.

Usage

Enable the AUTO_REPREPARE configuration parameter to reduce the number of reprepare operations that you must perform explicitly after modifying the schema of a table that is referenced by a dynamic SQL statement or a DML statement in an SPL routine.

For example, certain DDL statements modify the schema of a table, such as CREATE INDEX, DROP INDEX, DROP COLUMN, and RENAME COLUMN. If the AUTO_REPREPARE configuration parameter is disabled when these DDL statements are run, users might receive `-710` errors. These errors occur the next time that you run:

- An SPL routine that directly or indirectly references tables that were modified by the DDL statements
- A prepared object that references the tables that were modified by the DDL statements

Optimistic mode offers faster performance by not checking statements that successfully executed less than a second ago. In the unlikely event that tables were modified in the interim, some -710 errors might occur.

Set automatic repreparation on update statistics if you want to avoid the database server using an older, suboptimal execution plan.

Restriction:

Enabling AUTO_REPREPARE might have no effect on prepared statements or on SPL routines that reference tables in which DDL operations change the number of columns in the table, or change the data type of a column. After these schema changes, typically you must reissue the DESCRIBE statement, the PREPARE statement (for prepared objects), and the UPDATE STATISTICS FOR ROUTINE statement (for cursors associated with routines) for optimized execution plans of SPL routines that reference the table whose schema has been modified. Otherwise, the database server might issue SQL error `-710`.

AUTO_STAT_MODE configuration parameter

Use the AUTO_STAT_MODE configuration parameter to enable or disable the mode for selectively updating only stale or missing data distributions in UPDATE STATISTICS operations instead of updating statistics for all data distributions.

onconfig.std value

AUTO_STAT_MODE 1

Not set. If the AUTO_TUNE configuration parameter is set to 1, statistics are updated selectively.

values

`0` = Disables selective UPDATE STATISTICS operations.

`1` = Enables selective UPDATE STATISTICS operations.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

If an AUTO_STAT_MODE value is not set in your current `onconfig` file and you set the AUTO_TUNE configuration parameter.

Usage

When the AUTO_STAT_MODE configuration parameter or the AUTO_STAT_MODE session environment variable have enabled the automatic mode for selectively updating only stale or missing data distributions in UPDATE STATISTICS operations, the database server uses the value of the STATCHANGE configuration parameter to identify table or fragment distribution statistics that need to be updated.

In sessions where the AUTO_STAT_MODE configuration parameter and the AUTO_STAT_MODE session environment variable have different settings, the session environment variable takes precedence for the duration of that session, or until the AUTO_STAT_MODE session environment variable is reset.

AUTO_TUNE configuration parameter

Use the AUTO_TUNE configuration parameter to enable or disable all automatic tuning configuration parameters that have values that are not present in the `onconfig` file.

onconfig.std value

AUTO_TUNE 1

values

`0` = disabled

`1` = enabled

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If an individual automatic tuning configuration parameter is not set in your current `onconfig` file, the database server uses the value specified in the AUTO_TUNE configuration parameter for that configuration parameter.

The automatic tuning configuration parameters are:

- AUTO_AIOVPS
- AUTO_CKPTS
- AUTO_LRU_TUNING
- AUTO_READAHEAD
- AUTO_REPREPARE
- AUTO_STAT_MODE

If an automatic tuning configuration parameter is set in the current `onconfig` file, the database server uses the value that is in the `onconfig` file. The AUTO_TUNE configuration parameter does not change that value.

Your `onconfig` file is in the `%ONEDB_HOME%\etc` or `$ONEDB_HOME/etc` directory.

Example

Examples

Example 1: Suppose some of your automatic tuning configuration parameters are not set, but others have values:

AUTO_LRU_TUNING (value not set)
AUTO_STAT_MODE (value not set)
AUTO_LRU_CKPTS (value not set)
AUTO_AIOVPS 0
AUTO_REPREPARE 1
AUTO_READAHEAD 0

If you set the AUTO_TUNE configuration parameter to 1, the database server automatically changes the values that are not set to 1. The values that were previously set remain the same. The automatic tuning configuration parameters now have the following values:

AUTO_LRU_TUNING 1
AUTO_STAT_MODE 1
AUTO_CKPTS 1
AUTO_AIOVPS 0
AUTO_REPREPARE 1
AUTO_READAHEAD 0

Example 2: Suppose all of your automatic tuning configuration parameters are set and have the following values:

AUTO_LRU_TUNING 1
AUTO_STAT_MODE 1
AUTO_LRU_CKPTS 1
AUTO_AIOVPS 0
AUTO_REPREPARE 1
AUTO_READAHEAD 0

In this situation, the AUTO_TUNE configuration does not change any of the values.

Example 3: Suppose that you removed the automatic tuning configuration parameters from your `onconfig` file but now want to use them. You can set AUTO_TUNE to 1 to re-enable all of the automatic tuning configuration parameters.

AUTOLOCATE configuration parameter

Use the AUTOLOCATE configuration parameter to enable the automatic location of databases, indexes, and tables, and the automatic fragmentation of tables.

onconfig.std and default value

AUTOLOCATE 0

values

`0` = Disable automatic location and fragmentation.

`1 - 32` = Enable automatic location and fragmentation. The number indicates how many round-robin fragments to initially allocate to a table.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in memory and in your `onconfig` file by running the onmode -wf command.

When you reset the value dynamically in memory by running the onmode -wm command.

Usage

Use the AUTOLOCATE configuration parameter to control whether the database server controls the location of new databases, indexes, and tables and the fragmentation of those tables. If you set the AUTOLOCATE configuration parameter to a positive integer, the database server performs the following tasks:

- Stores new databases for which you do not specify a location in the optimal dbspace instead of in the root dbspace. By default, all dbspaces except dbspaces that are dedicated to tenant databases are available. However, you can control the list of available dbspaces.
- Fragments new tables by round-robin, where the number of fragments is equal to the value of the AUTOLOCATE configuration parameter.
- Adds more table fragments as the table grows.

If you set the value of the AUTOLOCATE configuration parameter to 0, new databases are created in the root dbspace by default. New tables and indexes are created in the same dbspace as the database and are not fragmented.

Automatic location is not applicable to tenant databases or the tables, fragments, and indexes within tenant databases.

You can override the automatic location of a database by specifying a dbspace with the IN clause in the CREATE DATABASE statement. Similarly, you can override the automatic location and fragmentation of a table by specifying a dbspace with the IN clause or a fragmentation strategy with the FRAGMENT BY clause in the CREATE TABLE statement.

When this configuration parameter is enabled, you can use the **autolocate database** arguments with the admin() or task() function to:

- Manage the list of dbspaces for automatic location and fragmentation. The list of available dbspaces is in the **sysautolocate** system catalog table.
- Disable automatic location and fragmentation for the specified database.

You can use the AUTOLOCATE environment option of the SET ENVIRONMENT statement of SQL to enable or disable the value of the AUTOLOCATE configuration parameter for a session.

BATCHEDREAD_INDEX configuration parameter

Use the BATCHEDREAD_INDEX configuration parameter to enable the optimizer to execute light scans for indexes. This reduces the number of times that a buffer is read, thus improving performance.

onconfig.std value

BATCHEDREAD_INDEX 1

values

`0` = Disable light scans for indexes.

`1` = Enable light scans for indexes.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

In sessions where the IFX_BATCHEDREAD_INDEX configuration parameter and the IFX_BATCHEDREAD_INDEX session environment variable have different settings, the session environment variable takes precedence for the duration of that session, or until the IFX_BATCHEDREAD_INDEX session environment variable is reset.

BATCHEDREAD_TABLE configuration parameter

Use the BATCHEDREAD_TABLE configuration parameter to enable or disable light scans on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data.

onconfig.std value

BATCHEDREAD_TABLE 1

values

`0` = Disable light scans on variable record-length tables

`1` = Enable light scans on variable record-length tables.

Compressed tables, and tables with rows longer than a page, are treated here as of variable record-length.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Except for compressed tables, tables with rows that are larger than a page, and tables of varying record length (such as VARCHAR, LVARCHAR, and NVARCHAR columns), the setting of BATCHEDREAD_TABLE has no effect on whether the query optimizer chooses a query execution path that includes a light scan.

The database server does not perform light scans on indexes, on system tables, nor on user tables whose rows include large objects with any of these storage attributes:

- blobspaces
- smartblob spaces
- partition blob.

You can use the IFX_BATCHEDREAD_TABLE environment option of the SET ENVIRONMENT statement to override the value of the BATCHEDREAD_TABLE configuration parameter for the current session.

BLOCKTIMEOUT configuration parameter

Use the BLOCKTIMEOUT configuration parameter to specify the number of seconds that a thread or database server will hang. After the timeout, the thread or database server will either continue processing or fail.

onconfig.std value

BLOCKTIMEOUT 3600

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

BTSCANNER Configuration Parameter

Use the BTSCANNER configuration parameter to set the B-tree scanner. The B-tree scanner improves transaction processing for logged databases when rows are deleted from a table with indexes. The B-tree scanner threads remove deleted index entries and rebalance the index nodes. The B-tree scanner automatically determines which index items are to be deleted.

onconfig.std value

BTSCANNER num=1,threshold=5000,rangesize=-1,alice=6,compression=default

range of values

See the Usage section.

separators

Use a comma between each field.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -C command.

After you run the SQL administration API task() or admin() function with the onmode and C arguments.

Usage

By default, the BTSCANNER configuration parameter starts one index cleaner thread, prioritizes cleaning indexes that have over 5000 deleted items, automatically adjusts the mode of index cleaning, and merges index pages at a level appropriate for indexes that have moderate growth and changes.

Syntax for the BTSCANNER configuration parameter

BTSCANNER [num= { 1 | *threads* } ,] [threshold=*thresh_size* ,] [rangesize=100 ,] [alice=*alice_mode* ,] [compression= { default | low | med | high }]

Table 60. Options for the BTSCANNER configuration parameter value

Field	Values
num	The *threads* value is a positive integer that sets the number of B-tree scanner threads to start at system startup. The default is `1`.
threshold	The *thresh_size* value is the minimum number of deleted items an index must encounter before an index is prioritized for cleaning. The default is `5000`.
rangesize	Specifies whether to allow leaf scans for small indexes: • -1 = Off. The alice mode is used for all index cleaning. • 100 = Small indexes are scanned by the leaf scan method.
alice	The *alice_mode* value controls index cleaning: • 0 = Off. • 1 = Uses exactly 8 bytes of memory. • 2 = Uses exactly 16 bytes of memory. • 3 - 12 = Default is 6. Sets the initial amount of memory that is used for index cleaning. Subsequently, the B-tree scanners automatically adjust the mode based on the efficiency of past cleaning operations.
compression	The level at which two partially used index pages are merged: • `low` = Use if you expect an index to grow quickly with frequent splits. • `med` or `default` = Default. Use if an index has moderate growth or changes. • `high` = Use if an index is 90 percent or more read-only or does not have many changes.

After all of the indexes above the threshold are cleaned, the indexes below the threshold are added to the prioritized list of indexes to be cleaned. Systems updated frequently should increase this value by a factor of 10 times or 100 times.

BUFFERPOOL configuration parameter

Use the BUFFERPOOL configuration parameter to configure how many data pages are cached in shared memory and how often those pages are flushed to disk between checkpoints. The default values of the BUFFERPOOL configuration parameter are adequate for many systems. However, you can change the values to tune the performance of your system.

onconfig.std values

Operating systems with 2 KB default page size:

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,
lru_max_dirty=60.50
BUFFERPOOL size=2k,buffers=50000,lrus=8,lru_min_dirty=50,
lru_max_dirty=60
```

Operating systems with 4 KB default page size:

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,
lru_max_dirty=60.50
BUFFERPOOL size=4k,buffers=10000,lrus=8,lru_min_dirty=50,
lru_max_dirty=60
```

default value if you created a server during installation

```
BUFFERPOOL default,memory='auto'
BUFFERPOOL size=page_size,memory=memory_size
```

The *page_size* value is the default page size. The initial size of the buffer pool is 32 MB. The maximum size, which is specified by the value of the memory field as either auto or the *memory_size* value, depends on the value of the AUTO_TUNE_SERVER_SIZE configuration parameter.

values

See the Usage section.

separators

Separate fields with a comma.

takes effect

After you edit your `onconfig` file and restart the database server.

When you add an entry dynamically in your `onconfig` file by running the onparams -b command.

When you add an entry dynamically by adding a dbspace with a different page size by running the onspaces -c -d command.

After you add an entry dynamically in your `onconfig` file by running the SQL administration API task() or admin() function with the add bufferpool argument.

Usage

Cached data pages are held in buffers. Buffers are contained in buffer pools. You need a buffer pool for each page size that you use for storage spaces. When the database server moves new data pages into shared memory, data pages that are the least-recently used are moved out of shared memory. The BUFFERPOOL configuration parameter controls the size of the buffer pool and how frequently data pages are flushed to disk. The BUFFERPOOL configuration parameter specifies the page size of the buffer pool, the number of buffers, the number of queues for least-recently used (LRU) data pages, and how frequently data pages in the LRU queues are flushed to disk.

The BUFFERPOOL configuration parameter has two entries in the `onconfig.std` file or in the `onconfig` file that was generated if you created a server during installation:

The BUFFERPOOL configuration parameter has two entries in the `onconfig.std` file:

- The first entry specifies the default values for a buffer pool for a dbspace with a non-default page size.
- The second entry specifies the default values for a buffer pool that is based on the default page size of the system.

The BUFFERPOOL configuration parameter entries that include the size field take precedence over the entry that includes the default field.

The BUFFERPOOL configuration parameter has two formats:

- Use the BUFFERPOOL configuration parameter with the memory field if you want to specify the size of your buffer pool in units of memory like MB or GB.
- Use the BUFFERPOOL configuration parameter with the buffers field if you want to specify the size of your buffer pool in units of pages, or to retain settings from a previous release.

You can use either format to enable the database server to expand the size of the buffer pool as needed to improve performance.

Restriction: You cannot combine formats in the `onconfig` file. All entries for the BUFFERPOOL configuration parameter in the `onconfig` file must have the same format or the database server does not start and the following error shows:

```
ERROR: Cannot mix buffer arguments with memory arguments.  (BUFFERPOOL)
```

The fields in the BUFFERPOOL entries are not case-sensitive and the fields can be listed in any order.

Syntax with the memory field

BUFFERPOOL { **default** | **size** = *page_size* **k** } [, **lrus** = *number_lrus*] [, **lru_min_dirty** = *min_percentage*] [, **lru_max_dirty** = *max_percentage*] [, **extendable** = { **0** | **1** [, **cache_hit_ratio** = *ratio*] }] [, **start_memory** = { **auto** | *start_size* [{ **kb** | **mb** | **gb** }] }] , **memory** = { **auto** | *max_size* [{ **kb** | **mb** | **gb** }] }

Syntax with the buffers field

BUFFERPOOL { **default** | **size** = *page_size* **[k]** } [, **lrus** = *number_lrus*] [, **lru_min_dirty** = *min_percentage*] [, **lru_max_dirty** = *max_percentage*] , **buffers** = *number_buffers* [, **extendable** = { **0** | **1** <extendable options> }]

extendable options

" [, **max_extends** = *extends*] "

" [, **next_buffers** = *number_buffers*] "

" [, **cache_hit_ratio** = *ratio*] "

Table 61. Options for the BUFFERPOOL configuration parameter value

Field	Values
buffers	Default is 1000. The *number_buffers* value is an integer >= 1000 that specifies the maximum number of shared-memory buffers. The maximum allowed number of buffers depends on the operating system, the bit size, and the page size: • UNIX™, 32-bit, with a 2 KB page size: 1000 - 1843200 • UNIX™, 32-bit, with a 4 KB page size: 1000 - 921600 • Windows™, 32-bit: 100 - 524288 • 64-bit: 100 - (2^{31}-1). For the actual value for your 64-bit platform, see your machine notes. For example, the maximum number of buffers on the Solaris platform is 536,870,912. Set the value of the buffers field to at least four buffers per user. If your system handles more than 500 concurrent users, specify at least 2000 buffers. Each buffer is the size of the operating system page. Therefore, the number of buffers that the database server requires depends on the amount of physical memory and how much memory is used by applications. For example, if the database server accesses 15 percent of the application data 90 percent of the time, allocate enough buffers to hold 15 percent of the data. Increasing the number of buffers can improve system performance. The number of buffers can have a significant affect on performance and use a large percentage of physical memory.

Table 61. Options for the BUFFERPOOL configuration parameter value

(continued)

Field	Values
	For more information, see The BUFFERPOOL configuration parameter and memory utilization on page .
cache_hit_ratio	Default is 90. The *ratio* value is an integer 0 - 100 that represents the threshold below which the buffer pool is extended. When the average read cache hit ratio remains below the value of *ratio* for approximately five minutes, the database server extends the buffer pool. The cache_hit_ratio field is valid only if extendable=1 is set.
extendable	Default is 1 if the memory field is set. Default is 0 if the buffers field is set. Whether the database server can extend the size of the buffer pool: • 0 = Disabled. The buffer pool cannot grow. • 1 = Enabled. The buffer pool can grow.
lru_max_dirty	Default is 60.00. The *max_percentage* value is a decimal number 0 - 100.00 that sets the percentage of modified pages in the LRU queues at which the queue is cleaned. This value is updated automatically as needed if the AUTO_LRU_TUNING configuration parameter is enabled.
lru_min_dirty	Default is 50.00. The *min_percentage* value is a decimal number 0 - 100.00 that sets the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory. Page cleaners might continue cleaning beyond the specified percentage under some circumstances. This value is updated automatically as needed if the AUTO_LRU_TUNING configuration parameter is enabled.
lrus	Default is 8. If the MULTIPROCESSOR configuration parameter is enabled, the default is the greater of 8 or the number of CPU VPs.

Table 61. Options for the BUFFERPOOL configuration parameter value

(continued)

Field	Values
	The *number_lrus* value is a positive integer that specifies the number of LRU (least recently used) queues in the buffer pool. The range of values depends on the bit size of the operating system: • 32-bit platforms: 8 - 128 • 64-bit platforms: 8 - 512 • 32-bit platforms: 1 - 128 • 64-bit platforms: 1 - 512 Set the value of lrus field to between four and the number of CPUs in your system. The more LRU queues that you specify, the more page cleaners work in parallel. However, setting the value of lrus field too high might result in excessive page-cleaner activity. The value of lrus field, in combination with the lru_min_dirty and lru_max_dirty fields control how frequently the shared-memory buffers are flushed to disk. For more information, see BUFFERPOOL and its effect on page cleaning on page .
max_extends	Default is 8. The *extends* value represents the maximum number of times that the database server can extend the buffer pool. The value of *extends* is 0 through the maximum number of segments, which depends on the operating system and bit size: • 32 bit = 16 • UNIX™ 64 bit = 24 • Windows™ 64 bit = 8 The max_extends field is valid only if buffers and extendable=1 are set.
memory	Default is auto. The *max_size* value represents the maximum size of the buffer pool. The range of values for *max_size* is:

Table 61. Options for the BUFFERPOOL configuration parameter value

(continued)

Field	Values
	• An integer that represents 32 MB - 4 TB. You can specify the size units of KB, MB, or GB. If you do not specify units, the default units are KB. • auto = The database server determines the maximum amount of shared memory to allocate to the buffer pool. The value of the AUTO_TUNE_SERVER_SIZE configuration parameter, if it is set, controls the maximum size of the buffer pool.
next_buffers	Default is 1000. The *number_buffers* value is an integer >= 1000 that specifies the number of shared-memory buffers by which the database server extends the buffer pool. The maximum value of *number_buffers* is limited by the amount of virtual shared memory. The *number_buffers* value is doubled every four extensions. The next_buffers field is valid only if buffers and extendable=1 are set.
size	The *page_size* value specifies the page size for buffers, in KB. The page size must be 2 - 16 KB and must be a multiple of the default page size. For example, if the default page size is 2 KB, the page size can be 2, 4, 6, 8, 10, 12, 14, or 16. If the default page size is 4 KB, the page size can be 4, 8, 12, or 16. The default value depends on the system default page size: • 2 KB default page size: `size=2k` • 4 KB default page size: `size=4k` The `k` is optional.
start_memory	Default is 32 MB. The *start_size* value represents the initial size of the buffer pool when the database server starts: • An integer that represents 32 MB through the maximum amount of shared memory that is available. You can specify the size units of KB, MB, or GB. If you do not specify units, the default units are KB. The initial size of the buffer pool might be larger than the value of *start_size* because the size must be a multiple of the size of a shared memory segment. • auto = The database server determines the initial amount of shared memory to allocate to the buffer pool.

Table 61. Options for the BUFFERPOOL configuration parameter value

(continued)

Field	Values
	If you do not set the start_memory field, the initial size of the buffer pool is equal to the value of the memory field. The start_memory field is valid only if the memory field is set.

The size of the buffer pool with the memory format

If you use the memory format, by default the buffer pool grows in size as needed. Shared memory segments are added to the buffer pool when the average cache read hit ratio is under the threshold. You can set the initial and maximum size of the buffer pool or allow the database server to determine the optimal sizes.

If the extendable field is set to 0, the buffer pool does not grow. The size is equal to the value of the start_memory field, if it is set, otherwise, the value of the memory field.

When you restart the server, the size of the buffer pool is reset to the value of the start_memory field.

The size of the buffer pool with the buffers format

If you use the buffers format, by default the buffer pool does not grow in size. The size is equal to the value of the buffers field.

If you set the extendable field to 1, shared memory segments are added to the buffer pool when the average cache read hit ratio is under the threshold. You must set the initial number of buffers in the buffers field. You can optionally set the number of buffers by which to extend the buffer pool, and the maximum number of times that the buffer pool can be extended, and the cache hit ratio. The number of buffers that are added to the buffer pool doubles every fourth extension.

Syntax for the BUFFERPOOL configuration parameter

`BUFFERPOOL` { `default` | `size` = *page_size* `k` } , `buffers` = *number_buffers* , `lrus` = *number_lrus* , `lru_min_dirty` = *min_percentage* , `lru_max_dirty` = *max_percentage*

Table 62. Options for the BUFFERPOOL configuration parameter value

Field	Values
size	The *page_size* value specifies the page size for buffers, in KB. The page size must be 2 - 16 KB and must be a multiple of the default page size. For example, if the default page size is 2 KB, the page size can be 2, 4, 6, 8, 10, 12, 14, or 16. If the default page size is 4 KB, the page size can be 4, 8, 12, or 16. The default value depends on the system default page size:

Table 62. Options for the BUFFERPOOL configuration parameter value

(continued)

Field	Values
	• 2K default page size: `size=2k` • 4K default page size: `size=4k` The `k` is optional.
buffers	The *number_buffers* value is an integer >= 100 that specifies the maximum number of shared-memory buffers. The maximum allowed number of buffers depends on the operating system, the bit size, and the page size: • UNIX™, 32-bit, with a 2K page size: 100 - 1843200 • UNIX™, 32-bit, with a 4K page size: 100 - 921600 • Windows™, 32-bit: 100 - 524288 • 64-bit: 100 - (2^{31}-1). For the actual value for your 64-bit platform, see your machine notes. For example, the maximum number of buffers on the Solaris platform is 536,870,912. Set the value of the buffers field to at least four buffers per user. If your system handles more than 500 concurrent users, specify at least 2000 buffers. Each buffer is the size of the operating system page. Therefore, the number of buffers that the database server requires depends on the amount of physical memory and how much memory is used by applications. For example, if the database server accesses 15 percent of the application data 90 percent of the time, allocate enough buffers to hold 15 percent of the data. Increasing the number of buffers can improve system performance. The number of buffers can have a significant affect on performance and use a large percentage of physical memory. For more information, see The BUFFERPOOL configuration parameter and memory utilization on page .
lrus	The *number_lrus* value is a positive integer that specifies the number of LRU (least recently used) queues in the buffer pool. The range of values depends on the bit size of the operating system: • 32-bit platforms: 1 - 128 • 64-bit platforms: 1 - 512 Set the value of lrus field to between four and the number of CPUs in your system. The more LRU queues you specify, the more page cleaners work in parallel. However, setting the value of lrus field too high might result in excessive page-cleaner activity.

Table 62. Options for the BUFFERPOOL configuration parameter value

(continued)

Field	Values
	The value of lrus field, in combination with the lru_min_dirty and lru_max_dirty fields control how frequently the shared-memory buffers are flushed to disk. For more information, see BUFFERPOOL and its effect on page cleaning on page .
lru_min_dirty	The *min_percentage* value is a decimal number 0 - 100 that sets the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory. The default value is 50.00. Page cleaners might continue cleaning beyond the specified percentage under some circumstances. This value is updated automatically as needed if the AUTO_LRU_TUNING configuration parameter is enabled.
lru_max_dirty	The *max_percentage* value is a decimal number 0 - 100 that sets the percentage of modified pages in the LRU queues at which the queue is cleaned. The default value is 60.50. This value is updated automatically as needed if the AUTO_LRU_TUNING configuration parameter is enabled.

Example

Example: Adding a BUFFERPOOL entry with the memory field

The following entry creates a buffer pool that has a 10 KB page size:

```
BUFFERPOOL size=10k,start_memory=auto,memory=4gb
```

The buffer pool is extendable up to 4 GB. The database server determines the initial size of the buffer pool and the sizes of extensions to the buffer pool.

Example

Example: Adding a BUFFERPOOL entry with the buffers field

The following entry creates a buffer pool that has a 2 KB page size:

```
BUFFERPOOL size=2k,extendable=1,buffers=1000,next_buffers=2000,max_extends=8
```

The buffer pool is extendable eight times. The buffer pool starts with 1000 buffers. The first three extensions to the buffer pool add 2000 buffers. The fourth through seventh extensions add 4000 buffers. The eighth extension adds 8000 buffers.

Example: Adding a BUFFERPOOL entry by adding a dbspace with a different page size

When you add a dbspace with a different page size with the onspaces utility, or when you add a buffer pool with the onparams utility, a BUFFERPOOL configuration parameter entry is added in the `onconfig` file. The following example shows a third entry:

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,lru_max_dirty=60.50
BUFFERPOOL size=2k,buffers=10000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
BUFFERPOOL size=6k
```

```
BUFFERPOOL default,buffers=10000,lrus=8,lru_min_dirty=50.00,lru_max_dirty=60.50
BUFFERPOOL size=2k,buffers=10000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
BUFFERPOOL size=6k,buffers=3000,lrus=8,lru_min_dirty=50,lru_max_dirty=60
```

When you create a dbspace with a non-default page size, the database server uses the existing BUFFERPOOL entry for that page size, if that entry exists. Otherwise, the database server uses the values from the BUFFERPOOL default line.

Buffer pools that are added while the database server is running are in virtual memory, not resident memory. Only those buffer pool entries that are specified in the `onconfig` file at startup are in resident memory.

CHECKALLDOMAINSFORUSER configuration parameter

Use the CHECKALLDOMAINSFORUSER configuration parameter to check all of the domains for all users.

onconfig.std value

Not in the `onconfig.std` file

values

`0` = Disabled

`1` = Enabled

takes effect

After you edit your `onconfig` file and restart the database server.

CKPTINTVL configuration parameter

Use the CKPTINTVL configuration parameter to specify the frequency, expressed in seconds, at which the database server checks to determine whether a checkpoint is needed. When a checkpoint occurs, all pages in the shared-memory buffer pool are written to disk.

onconfig.std value

CKPTINTVL 300

values

Any value greater than or equal to 0

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The RTO_SERVER_RESTART and CKPTINTVL configuration parameters are mutually exclusive. If the RTO_SERVER_RESTART configuration parameter is enabled, it will trigger checkpoints and CKPTINTVL values are ignored. Otherwise, CKPTINTVL values are used to trigger checkpoints.

If you set the CKPTINTVL configuration parameter to an interval that is too short, the system spends too much time performing checkpoints, and the performance of other work suffers. If you set the CKPTINTVL configuration parameter to an interval that is too long, fast recovery might take too long.

In practice, 30 seconds is the smallest interval that the database server checks. If you specify a checkpoint interval of 0, the database server does not check if the checkpoint interval has elapsed. However, the database server still performs checkpoints. Other conditions, such as the physical log becoming 75 percent full, also cause the database server to perform checkpoints.

CLEANERS configuration parameter

Use the CLEANERS configuration parameter to specify the number of page-cleaner threads available during the database server operation. By default, the database server always runs one page-cleaner thread. A general guideline is one page cleaner per disk drive. The value specified has no effect on the size of shared memory.

Based on the server work load, the server automatically attempts to optimize AIO VPs and page-cleaner threads and adjust the number of AIO VPs and page-cleaner threads upward when needed. Automatic AIO VP and page-cleaner thread tuning can be disabled using the environmental variable IFX_NO_AIOVP_TUNING or the onmode -wm utility option.

onconfig.std value

CLEANERS 8

values

1 - 128

units

Number of page-cleaner threads

takes effect

After you edit your `onconfig` file and restart the database server.

CLUSTER_TXN_SCOPE configuration parameter

Set the CLUSTER_TXN_SCOPE configuration parameter to configure your high-availability cluster so that when a client session issues a commit, the server blocks the session until the transaction is applied in that session, on a secondary server, or across the cluster.

onconfig.std value

CLUSTER_TXN_SCOPE SERVER

values

- `SESSION` = When a client session issues a commit, the database server blocks the session until the effects of the transaction commit are returned to that session. After control is returned to the session, other sessions at the same database server or on other database servers in the cluster might be unaware of the transaction commit and the transaction's effects.
- `SERVER` (default behavior) = When a client session issues a commit, the database server blocks the session until the transaction is applied at the database server from which the client session issued the commit. Other sessions at that database server are aware of the transaction commit and the transaction's effects. Sessions at other database servers in the cluster might be unaware of the transaction's commit and its effects. This behavior is default for high-availability cluster servers.
- `CLUSTER` = When a client session issues a commit, the database server blocks the session until the transaction is applied at all database servers in the high-availability cluster, excluding RS secondary servers that are using DELAY_APPLY or STOP_APPLY. Other sessions at any database server in the high-availability cluster, excluding RS secondary servers that are using DELAY_APPLY or STOP_APPLY, are aware of the transaction commit and the transaction's effects.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

After you run the SQL administration API task() or admin() function with the -wf CLUSTER_TXN_SCOPE=*value* or -wm CLUSTER_TXN_SCOPE=*value* arguments.

Usage

Set the CLUSTER_TXN_SCOPE configuration parameter to control transaction-commit returns from a high-availability cluster to client applications. Cluster transaction coordination can delay the returning of a transaction commit to a client application until the transaction is applied to a secondary-server or all secondary servers in a high-availability cluster. This process prevents operation failures due to asynchronous log processing, and ensures that the steps of multistep processes occur in serial order.

Cluster transaction coordination does not apply to RS secondary servers that have a DELAY_APPLY or STOP_APPLY configuration parameter value other than `0`. Transactions do not need to be applied on the RS secondary servers before client applications can receive commits.

CLUSTER_TXN_SCOPE affects sessions on read-only secondary servers and updatable secondary servers.

Before HCL OneDB™ version 11.70.xC6, high-availability cluster servers had the following default behaviors:

- Primary servers had a cluster transaction scope of SERVER.
- Read-only secondary servers were in the dirty-read isolation level, and could read uncommitted data.
- Updatable secondary servers had a cluster transaction scope of SESSION.

Example

Example 1: Transactions coordination between high-availability cluster servers

In this example, a client application starts a two-step process. The client application inserts data on the primary database server, and then starts processing of the data on an HDR secondary server.

If a SELECT on the inserted data is attempted on the HDR secondary server before the logs from the primary server are applied on the HDR secondary server, the operation fails. To prevent this failure, set the primary server's CLUSTER_TXN_SCOPE configuration parameter to `CLUSTER`, so that the client application does not receive a commit, and cannot start data processing, until the data insertion is also applied on the HDR secondary server.

Example

Example 2: Transaction coordination on a database server

In this example, you have a client application that is divided into several stages of processing. Each stage of processing uses a different SQL session to connect to the database server. The application updates data, and then another part of the application processes the updated data in a different SQL session.

If CLUSTER_TXN_SCOPE is set to SESSION, the part of the application that processes the updated data might not be aware of an update's results and a failure can occur. To prevent this failure, set the database server's CLUSTER_TXN_SCOPE configuration parameter to `SERVER`, so that the client application does not receive a commit, and cannot start data processing until the update completes on the database server.

CONSOLE configuration parameter

Use the CONSOLE configuration parameter to specify the path and name for console-message file.

onconfig.std values

On UNIX™: `$ONEDB_HOME/tmp/online.con`

On Windows™: `online.con`

values

pathname = Full path name of the `online.con` file.

takes effect

After you edit your `onconfig` file and restart the database server.

CONVERSION_GUARD configuration parameter

Use the CONVERSION_GUARD configuration parameter to specify whether HCL OneDB™ stops or continues an upgrade to a new version of the server if an error occurs during the upgrade process.

onconfig.std value

CONVERSION_GUARD 2

values

`0` = Disabled.

`1` = Enable a restore point as part of the upgrade process, and stop the upgrade if an error related to capturing restore point data occurs.

`2` = Enable a restore point as part of the upgrade process, and continue the upgrade even if an error related to capturing restore point data occurs.

units

Integer

takes effect

When the database server is restarted

Usage

By default:

- The CONVERSION_GUARD configuration parameter is on (set to 2). If an upgrade to the new version of the server fails, you can use the onrestorept utility to restore your data.
- The server stores the restore point data in the `$ONEDB_HOME/tmp` directory.

If the CONVERSION_GUARD configuration parameter is set to 1 or 2 and the upgrade to the new version of the server fails, you can use the onrestorept utility to restore your data.

If the CONVERSION_GUARD configuration parameter is set to 2 and conversion guard operations fail (for example, because the server has insufficient space to store restore point data), and the upgrade fails, you cannot use the onrestorept utility to restore your data.

You can change the value of the CONVERSION_GUARD configuration parameter or change the directory specified in the RESTORE_POINT_DIR configuration parameter before starting the server that initiates an upgrade to a new version of the server. You cannot change the CONVERSION_GUARD or RESTORE_POINT_DIR values during an upgrade.

DATASKIP Configuration Parameter

Use the DATASKIP configuration parameter to control whether the database server skips a dbspace that is unavailable during the processing of a transaction.

onconfig.std value

Not set. No dbspaces are skipped.

values

See the Usage section.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onspaces -f command.

After you run the SQL administration API task() or admin() function with the set dataskip argument.

Usage

Whenever the database server skips over a dbspace during query processing, a warning is returned.

Enable the DATASKIP configuration parameter with caution because the results are always suspect. Only enable the parameter in the following situations:

- You can accept the compromised integrity of transactions.
- You can determine that the integrity of the transaction is not compromised, which can be difficult and time consuming.

Syntax for the DATASKIP configuration parameter

`DATASKIP` { `ALL` | `OFF` | `ON` *dbspace_name* }

Table 63. Options for the DATASKIP configuration parameter value

Field	Description
ALL	Skip all unavailable fragments.
OFF	All fragments, including unavailable fragments, are processed.
ON	The *dbspace_name* value specifies one or more dbspaces to skip, separated by commas.

An application can use the SQL statement SET DATASKIP to override the value of the DATASKIP configuration parameter.

The previously reserved SQLCA warning flag **sqlwarn.sqlwarn7** is set to `W` for .

DBCREATE_PERMISSION configuration parameter

Use the DBCREATE_PERMISSION configuration parameter to restrict the permission to create databases to the user that you specify.

The **informix** user always has permission to create databases. To restrict the ability to create databases to the **informix** user, set the DBCREATE_PERMISSION configuration parameter to **informix**.

onconfig.std value

#DBCREATE_PERMISSION informix

On UNIX™: Not set. Any user can create databases.

On Windows™: #DBCREATE_PERMISSION informix

default value

Any user can create databases.

units

user names

separator

Comma. You can also include multiple copies of the DBCREATE_PERMISSION configuration parameter in the `onconfig` file to give more users permission to create databases.

takes effect

After you edit your `onconfig` file and restart the database server.

The DBCREATE_PERMISSION configuration parameter does not provide permissions to create tenant databases. Users must have the TENANT privilege to create tenant databases. Grant the TENANT privilege by running the admin() or task() SQL administration API function with the **grant admin** argument.

DB_LIBRARY_PATH configuration parameter

Use the DB_LIBRARY_PATH configuration parameter to specify a comma-separated list of valid directory prefix locations from which the database server can load external modules, such as DataBlade® modules. You can also include server environment variables, such as $ONEDB_HOME, in the list.

You must specify the paths to the external modules exactly as the paths are registered with the database server. Relative paths or paths that include double periods (..) are not valid. External modules in the file systems that are not specified by this parameter cannot be loaded. This list is scanned prior to loading C language modules.

If you set this configuration parameter, you must also include the string `$ONEDB_HOME/extend` as part of the value. If the string `$ONEDB_HOME/extend` is not included in DB_LIBRARY_PATH, built-in extensions, DataBlade® modules, and the BladeManager utility do not load.

onconfig.std value

Not set

if not present

The database server can load external modules from any location

values

List of path names (up to 512 bytes)

separators

Comma

takes effect

After you edit your `onconfig` file and restart the database server.

DBSERVERALIASES configuration parameter

Use the DBSERVERALIASES configuration parameter to specify an alias name, or a list of unique alias names for the database server. Each alias defined by the DBSERVERALIASES configuration parameter can be used in a different connection, as specified by entries in the `sqlhosts` information.

onconfig.std value

Not set. No aliases are defined.

values

One to 32 alias names, separated by commas. Each alias name can be optionally followed by a minus sign and an integer from 1 - 50 that specifies the number of multiple listener threads to use for the **onimcsoc** or **onsoctcp** protocols. For example, the following two alias names each have four listener threads: `alias_a-4,alias_b-4`. The listener thread number is ignored for other protocols.

The maximum length of an alias is 128 bytes. Additional aliases beyond 32 are ignored. The maximum length of a DBSERVERALIASES entry is 512 bytes. You can include multiple lines of DBSERVERALIASES configuration parameters in the `onconfig` file.

An alias name must begin with a letter and can include any printable character, except the following:

- Uppercase characters
- A field delimiter (blank space or tab)
- A newline character
- A comment character (#)
- A hyphen or minus (= ASCII 45) character
- The @ character
- A blank space

separators

Separate entries with a comma. Do not include blank spaces.

takes effect

After you edit your `onconfig` file and restart the database server and update the `sqlhosts` information of each database server.

Usage

If HCL OneDB™ supports more than one communication protocol (for example, both an IPC mechanism and the TCP network protocol), you must describe each valid connection to the database server with an entry in the `sqlhosts` information. For example, suppose you have a server that has the name `sanfrancisco` defined by the DBSERVERNAME configuration parameter setting, and you set a DBSERVERALIASES value of `menlo` for different connection. You must specify information for both of the `sanfrancisco` and `menlo` servers in the `sqlhosts` information. Similarly, if the database server needs to support both the standard HCL OneDB™ protocols and the Distributed Relational Database Architecture™ (DRDA®) protocols, assign an alias to the DRDA® database server and add an entry for this alias in the `sqlhosts` file.

For each alias listed in the DBSERVERALIASES configuration parameter, the database server starts an additional listener thread. If you have many client applications connecting to the database server, you can distribute the connection requests between several listener threads and speed connection times. To take advantage of the alternate connections, program some of your client applications to connect to a database server alias name instead of the database server name.

High-availability cluster servers that use shared-memory connections must also have TCP connection aliases for server-to-server communication. If a high-availability cluster server's DBSERVERNAME is associated with a shared-memory `sqlhosts` file entry, you must create a TCP alias for the server by setting a DBSERVERALIASES value, setting the HA_ALIAS configuration parameter to the DBSERVERALIASES value, and then creating a TCP `sqlhost` file entry for the alias.

Note: Service name used for loopback replication group should be added to DBSERVERALIAS list, and it should appear after service name used for primary Enterprise Replication group.

DBSERVERNAME configuration parameter

Use the DBSERVERNAME configuration parameter to specify a unique name that you want to associate with the database server. You specify this configuration parameter when you install the database server.

onconfig.std value

Not set. A database server name is not defined.

if not present

On UNIX™: *hostname*

On Windows™: ol_*hostname*

The *hostname* variable is the name of the host computer.

values

A database server name that has a maximum length of 128 bytes.The database server name can be optionally followed by a minus sign and an integer from 1 - 50 that specifies the number of multiple listener threads to use for the **onimcsoc** or **onsoctcp** protocols. The default number of listener threads is 1. For example, the following database server name has four listener threads: `ifxserver-4`. The listener thread number is ignored for other protocols.

A database server name must begin with a letter and can include any printable character, except the following:

- Uppercase characters
- A field delimiter (blank space or tab)
- A newline character
- A comment character (#)
- A hyphen or minus (= ASCII 45) character
- The @ character
- A blank space

takes effect

After you edit your `onconfig` file and restart the database server and update the `sqlhosts` file or registry of each database server. In addition, the **ONEDB_SERVER** environment variable for all users might need to be changed.

Usage

The database server name is associated with a communication protocol that is specified in the `sqlhosts` file or registry. If the database server uses multiple communication protocols, define values for database server names with the DBSERVERALIASES configuration parameter.

Client applications use the database server name in the **ONEDB_SERVER** environment variable and in SQL statements such as CONNECT and DATABASE, which establish a connection to a database server.

Important: To avoid conflict with other instances of HCL OneDB™ database servers on the same computer or node, you should use the DBSERVERNAME configuration parameter to assign a database server name explicitly.

High-availability cluster servers that use shared-memory connections must also have TCP connection aliases for server-to-server communication. If a high-availability cluster server's DBSERVERNAME is associated with a shared-memory `sqlhosts` file entry, you must create a TCP alias for the server by setting a DBSERVERALIASES value, setting the HA_ALIAS configuration parameter to the DBSERVERALIASES value, and then creating a TCP `sqlhost` file entry for the alias.

DBSPACETEMP configuration parameter

Use the DBSPACETEMP configuration parameter to specify a list of dbspaces that the database server uses to globally manage the storage of temporary tables.

DBSPACETEMP improves performance by enabling the database server to spread out I/O for temporary tables efficiently across multiple disks. The database server also uses temporary dbspaces during backups to store the before-images of data that are overwritten while the backup is occurring.

onconfig.std value

Not set. Temporary tables are stored in the root dbspace.

separators

Comma or colon (no white space)

values

One or more dbspace names. Dbspaces can be standard dbspace, temporary dbspaces, or both. Separate dbspace names with a colon or comma. The length of the list cannot exceed 254 bytes.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

DBSPACETEMP can contain dbspaces with a non-default page size and dbspaces in the DBSPACETEMP list can have different page sizes.

If a client application needs to specify an alternative list of dbspaces to use for its temporary-table locations, the client can use the **DBSPACETEMP** environment variable to list them. The database server uses the storage locations that the **DBSPACETEMP** environment variable specifies only when you use the HIGH option of UPDATE STATISTICS.

If both standard and temporary dbspaces are listed in the DBSPACETEMP configuration parameter or environment variable, the following rules apply:

- Sort, backup, implicit, and nonlogging explicit temporary tables are created in temporary dbspaces if adequate space exists.
- Explicit temporary tables created without the WITH NO LOG option are created in standard (rather than temporary) dbspaces.

When you create a temporary dbspace with the onspaces utility or the SQL administration API admin() or task() function, the database server does not use the newly created temporary dbspace until you modify the DBSPACETEMP configuration parameter to include the new temporary dbspace or set the DBSPACETEMP environment variable and restart the session.

Note: The DBSPACETEMP configuration parameter may be modified dynamically with the onmode -wf or onmode -wm commands.

When set in a session, the DBSPACETEMP environment variable overrides the DBSPACETEMP configuration parameter.

Use Hash Join Overflow and DBSPACETEMP

HCL OneDB™ uses an operating-system directory or file to direct any overflow that results from certain database operations, if you do not set the **DBSPACETEMP** environment variable or DBSPACETEMP configuration parameter.

You can specify the operating-system directory or file in the following ways:

- SELECT statement with GROUP BY clause
- SELECT statement with ORDER BY clause
- Hash-join operation
- Nested-loop join operation
- Index builds

Location of the sort overflow files

The following table lists the environment variables and ONCONFIG configuration parameters that you can use to specify the location of the sort overflow files.

Table 64. Location of sort overflow files

Variable or Parameter	Location of the sort overflow files
PSORT_DBTEMP environment variable	The location specified in the environment variable
DBSPACETEMP environment variable	The location specified in the environment variable
DBSPACETEMP configuration parameter specified in the ONCONFIG file	The dbspace that is specified in the ONCONFIG file DBSPACETEMP configuration parameter

If more than one variable or parameter is specified, the priority by which the HCL OneDB™ determines the location of the sort overflow files is:

1. PSORT_DBTEMP environment variable
2. DBSPACETEMP environment variable
3. DBSPACETEMP ONCONFIG variable
4. DUMPDIR
5. $ONEDB_HOME/tmp

If the environment variables or configuration parameter are not set, the sort overflow files are placed in the **$ONEDB_HOME/tmp** directory and the temporary tables are placed in the rootdbspace.

DD_HASHMAX configuration parameter

Use the DD_HASHMAX configuration parameter to specify the maximum number of tables in each hash bucket in the data-dictionary cache.

A *hash bucket* is the unit of storage (typically a page) whose address is computed by the hash function. A hash bucket contains several records.

For example, if the DD_HASHMAX configuration parameter is set to 10 and the DD_HASHSIZE configuration parameter is set to 59, you can store information about 590 tables in the data-dictionary cache, and each hash bucket can have a maximum of 10 tables.

Use a text editor to modify the configuration file.

onconfig.std value

DD_HASHMAX 10

values

Positive integers

units

Maximum number of tables in a hash bucket

takes effect

After you edit your `onconfig` file and restart the database server.

DD_HASHSIZE configuration parameter

Use the DD_HASHSIZE configuration parameter to specify the number of hash buckets or lists that are in the data-dictionary cache.

Use a text editor to modify the configuration file.

onconfig.std value

DD_HASHSIZE 31

values

Any positive integer; a prime number is recommended

units

Number of hash buckets or lists

takes effect

After you edit your `onconfig` file and restart the database server.

DEADLOCK_TIMEOUT configuration parameter

Use the DEADLOCK_TIMEOUT configuration parameter to specify the maximum number of seconds that a database server thread can wait to acquire a lock.

Use this parameter only for distributed queries that involve a remote database server. Do not use this parameter for nondistributed queries.

onconfig.std value

DEADLOCK_TIMEOUT 60

values

Positive integers

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If a distributed transaction is forced to wait longer than the number of seconds specified with the DEADLOCK_TIMEOUT configuration parameter, the thread that owns the transaction assumes that a multi-server deadlock exists.

DEF_TABLE_LOCKMODE configuration parameter

Use the DEF_TABLE_LOCKMODE configuration parameter to specify the lock mode at the page or row level for new tables.

onconfig.std value

`PAGE`

values

`PAGE` = sets lock mode to page for new tables

`ROW` = sets lock mode to row for new tables

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

precedence rules

You can supersede all other lock mode settings for a specific table by including the LOCK MODE clause in the CREATE TABLE or ALTER TABLE statement.

The **IFX_DEF_TABLE_LOCKMODE** environment variable set on the client takes precedence over the variable on the server and the DEF_TABLE_LOCKMODE configuration parameter.

The **IFX_DEF_TABLE_LOCKMODE** environment variable set on the server takes precedence over the DEF_TABLE_LOCKMODE configuration parameter.

Usage

If the DEF_TABLE_LOCKMODE configuration parameter is set to `ROW`, it sets the lock mode to row for every newly created table for all sessions that are connected to logging or nonlogging databases. This parameter has no effect on the lock mode for existing tables.

If the DEF_TABLE_LOCKMODE configuration parameter is set to PAGE, the USELASTCOMMITTED configuration parameter and COMMITTED READ LAST COMMITTED option of the SET ISOLATION statement cannot enable access to the most recently committed data in tables on which uncommitted transactions hold exclusive locks, unless the tables were explicitly created or altered to have ROW as their locking granularity.

DEFAULTESCCHAR configuration parameter

The DEFAULTESCCHAR configuration parameter specifies the default escape character that is used for LIKE and MATCHES conditions.

onconfig.std value

DEFAULTESCCHAR backslash character (\ ).

if not present

The backslash character (\ ) is used if no value is set in the `onconfig` file.

values

`\` = The backslash character is used as the escape character.

NONE = No default escape character.

character = Any one-character value can be used as the escape character.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The default value can be overridden in a session by using the `SET ENVIRONMENT DEFAULTESCCHAR` statement with the escape character that you want to use. For example:

```
SET ENVIRONMENT DEFAULTESCCHAR '\'
```

DELAY_APPLY Configuration Parameter

Use the DELAY_APPLY configuration parameter to configure RS secondary servers to wait for a specified period of time before applying logs.

onconfig.std value

DELAY_APPLY 0

default value

0

values

0 = Apply logs

-1= Stage log files and apply data immediately.

A number followed by a time unit: for example, `1H` sets the delay to one hour.

number: `-1`-`999` = Number of days, minutes, hours, or seconds to wait.

time_unit: `D`, `H`, `M`, or `S`, where *D* = Days, *H* = Hours, *M* = Minutes, and *S* = Seconds. Values are not case sensitive.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Delaying the application of log files allows you to recover quickly from erroneous database modifications by restoring the data from the RS secondary server. When setting the value of DELAY_APPLY you must also set LOG_STAGING_DIR. If DELAY_APPLY is configured and LOG_STAGING_DIR is not set to a valid and secure directory, then the server cannot be initialized.

You must specify a valid and secure location for the log files by setting the LOG_STAGING_DIR configuration parameter. The logs in the staging directory are purged after the last checkpoint has been processed on the RS secondary server.

To see information about the data being sent to the log-staging directory set for a RS secondary server, run the onstat -g rss verbose command on the RS secondary server.

If the write to the staging file fails, the RS secondary server raises event alarm 40007.

If a remote stand-alone secondary (RSS) server has its DELAY_APPLY configuration parameter set to a value other than 0, that server cannot use cluster transaction coordination.

DIRECT_IO configuration parameter (UNIX™)

Use the DIRECT_IO configuration parameter to control the use of direct I/O for cooked files used for dbspace chunks.

This parameter enables direct I/O (bypassing file system buffering) on UNIX™ platforms or concurrent IO (bypassing both file system buffering and unnecessary write serialization) on AIX® operating systems.

onconfig.std value

DIRECT_IO 0

values

`0` = Neither direct I/O or concurrent I/O is used

`1` = Direct I/O, which bypasses file system buffering, is used if available

`2` = Concurrent I/O is enabled on AIX® operating systems (The concurrent I/O option includes direct I/O and concurrent I/O.)

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Direct I/O can only be used for dbspace chunks whose file systems support direct I/O for the page size.

By using direct I/O, you might be able to reduce the number of AIO virtual processors.

If direct I/O is enabled, KAIO (kernel asynchronous I/O) is used if the file system supports it. However, KAIO is not used if the environment variable KAIOOFF is set. When direct IO and KAIO are both used, the number of AIO virtual processors can be reduced. If direct IO is used, but KAIO is not, the number of AIO virtual processors should not be reduced.

HCL OneDB™ does not use direct or concurrent I/O for cooked files used for temporary dbspace chunks.

On AIX®, if HCL OneDB™ uses concurrent I/O for a chunk, another program (such as an online external backup program) must also use concurrent I/O. If not, the file open operation will fail.

If HCL OneDB™ uses direct I/O for a chunk, and another program tries to open the chunk file without using direct I/O, the open operation will normally succeed, but there can be a performance penalty. The penalty can occur because the file system might attempt to ensure that each open operation views the same file data, either by not using direct I/O at all for the duration of the conflicting open operation, or by flushing the file system cache before each direct I/O and invalidating the file system cache after each direct write.

Direct I/O is used for dbspace chunks on Windows™ platforms regardless of the value of the DIRECT_IO configuration parameter.

DIRECTIVES configuration parameter

Use the DIRECTIVES configuration parameter to enable or disable the use of optimizer directives. These directives specify behavior for the query optimizer in developing query plans for SELECT, UPDATE, and DELETE statements.

onconfig.std value

DIRECTIVES 1

values

`0` = Optimizer directives disabled

`1` = Optimizer directives enabled

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

environment variable

IFX_DIRECTIVES

Usage

Set DIRECTIVES to `1`, which is the default value, to enable the database server to process optimizer directives. Set DIRECTIVES to `0` to disable the database server from processing directives.

Client programs also can set the **IFX_DIRECTIVES** environment variable to `ON` or `OFF` to enable or disable processing of directives by the database server. The setting of the **IFX_DIRECTIVES** environment variable overrides the setting of the DIRECTIVES configuration parameter. If you do not set the **IFX_DIRECTIVES** environment variable, all sessions for a client inherit the database server configuration for processing directives.

DISABLE_B162428_XA_FIX configuration parameter

Use the DISABLE_B162428_XA_FIX configuration parameter to specify when transactions are freed.

onconfig.std value

Not in the `onconfig.std` file.

values

`0` = (Default) Frees transactions only when an xa_rollback is called.

`1` = Frees transactions if transaction rollback for other than an xa_rollback.

units

Integer

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Set DISABLE_B162428_XA_FIX to 1 to immediately free all global transactions after a transaction rollback, which is the default for HCL OneDB™ 9.40 and earlier versions. The default behavior for HCL OneDB™ 10.0 is to free global transactions after an xa_rollback is called, and this behavior is required to confirm to the XA state table that a transaction can be freed only after xa_rollback is called. Setting DISABLE_B162428_XA_FIX to 1 ensures that applications written for the earlier version of HCL OneDB™ work properly.

You can override the DISABLE_B162428_XA_FIX configuration parameter for a client session with the **IFX_XASTDCOMPLIANCE_XAEND** environment variable. Setting **IFX_XASTDCOMPLIANCE_XAEND** to 1 will free transactions only when an xa_rollback is called. Setting **IFX_XASTDCOMPLIANCE_XAEND** to 0 will free transactions if the transaction rollback is for other than an xa_rollback.

DISK_ENCRYPTION configuration parameter

The DISK_ENCRYPTION configuration parameter controls the encryption of storage spaces.

onconfig.std value

Not set. Storage space encryption is disabled.

values

See Usage section.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Use the DISK_ENCRYPTION configuration parameter to enable storage space encryption, set the name of the encryption file names, and specify the encryption cipher. Any storage spaces that you create after you set the DISK_ENCRYPTION configuration parameter are encrypted by default. Storage spaces that you created before you set the DISK_ENCRYPTION configuration parameter are not automatically encrypted. When storage space encryption is enabled, you can restore a storage space as encrypted or unencrypted, regardless of whether the space was encrypted at the time of the back up.

Syntax for the DISK_ENCRYPTION configuration parameter

DISK_ENCRYPTION keystore = *keystore_name* [, **cipher** = { **aes128** | **aes192** | **aes256** }] [, **rollfwd_create_dbs** = { **encrypt** | **decrypt** }]

Table 65. Options for the DISK_ENCRYPTION configuration parameter value

Field	Value
keystore	The *keystore* specifies the name of the keystore and stash file names. The files are created in the `ONEDB_HOME/etc` directory: • *`keystore`*.p12 = The keystore file that contains the security certificates. • *`keystore`*`.sth` = The stash file that contains the encryption password.

Table 65. Options for the DISK_ENCRYPTION configuration parameter value (continued)

Field	Value
	You must manually back up the keystore and password stash files. These files are not backed up when you run a back up with the ON-Bar utility.
cipher	Specifies the encryption cipher: • aes128 = Default. Advanced Encryption Standard cipher with 128-bit keys. • aes192 = Advanced Encryption Standard cipher with 192-bit keys. • aes256 = Advanced Encryption Standard cipher with 256-bit keys.
rollfwd_create_dbs	Specifies whether to encrypt a storage space that is created by the rolling forward of the logical log during a restore: • encrypt = Encrypt the newly created storage space • decrypt = Do not encrypt the newly created storage space By default, storage spaces that are created by the rolling forward of the logical log have the same encryption state as the original storage space.

DRDA_COMMBUFFSIZE configuration parameter

Use the DRDA_COMMBUFFSIZE configuration parameter to specify the size of the DRDA® communications buffer.

When a DRDA® session is established, the session is allocated a communication buffer equal to the current buffer size. If the buffer size is subsequently changed, existing connections are not affected, but new DRDA® connections use the new size. HCL OneDB™ silently resets values greater than 2 Megabyte to 2 Megabytes and resets values less than 4 Kilobytes to the 32 Kilobyte default value.

onconfig.std value

Not in the `onconfig.std` file.

if not present

32K

values

Minimum = 4 Kilobytes

Maximum = 2 Megabytes

takes effect

When shared memory is initialized

Usage

Users might specify the DRDA_COMMBUFFSIZE value in either MB or KB by adding either 'M' or 'K' to the value. The letter is not case sensitive, and the default is kilobytes. For example, a one megabyte buffer can be specified in any of these ways:

- DRDA_COMMBUFFSIZE 1M
- DRDA_COMMBUFFSIZE 1m
- DRDA_COMMBUFFSIZE 1024K
- DRDA_COMMBUFFSIZE 1024k
- DRDA_COMMBUFFSIZE 1024

DRAUTO configuration parameter

Set the DRAUTO configuration parameter to specify a HDR-failover method for HDR high-availability systems.

onconfig.std value

DRAUTO 0

Range of values

Value	Description
0	Automatic failover is disabled. When the primary server fails or loses network connectivity, the HDR secondary server becomes read-only.
1	Automatic failover is enabled. When the primary server fails or loses network connectivity, convert the HDR secondary server to a standard server. The HDR secondary server gracefully ends client connections and shuts down, and then restarts as a standard server. When the failed primary server restarts or reconnects to the network, convert the standard server back to the HDR secondary server. Do not use this setting if you configured Connection Managers to perform failover.
2	Automatic failover is enabled. When the primary server fails or loses network connectivity, convert the HDR secondary server to a primary server. The HDR secondary server maintains client connections and does not shut down. When the failed primary server restarts or reconnects to the network, convert it to an HDR secondary server. Do not use this setting if you configured Connection Managers to perform failover.

Value	Description
3	Failover is controlled by Connection Managers. Connection Managers must be configured and active for automatic failover.

Takes effect

When shared memory is initialized.

Usage

All servers of a high-availability cluster must have the same DRAUTO configuration parameter setting.

The DRAUTO configuration parameter does not control failover for SDS secondary servers or RS secondary servers. To set automatic failover for a high-availability cluster that has SD secondary servers or RS secondary servers, configure Connection Managers.

Important: If you are using Connection Managers to control failover, the DRAUTO configuration parameter must be set to 3 on all cluster servers. You must not perform a manual failover while Connection Managers are active.

DRIDXAUTO configuration parameter

Use the DRIDXAUTO configuration parameter to specify whether the primary High-Availability Data Replication (HDR) server automatically starts index replication if the secondary HDR server detects a corrupted index.

onconfig.std value

DRIDXAUTO 0

values

`0` = Off

`1` = On

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

To alter the value of the DRIDXAUTO configuration parameter for an active server instance, use the onmode -d idxauto command. You do not need to restart the server instance. However, the onmode -d idxauto command will not change the value of the DRIDXAUTO configuration parameter in the `onconfig` file.

DRINTERVAL configuration parameter

Use the DRINTERVAL configuration parameter to specify the maximum number of seconds between flushes of the data-replication buffer, whether to use HDR SYNC mode, or whether to use the synchronization mode that is specified by the HDR_TXN_SCOPE configuration parameter.

onconfig.std value

DRINTERVAL 30

DRINTERVAL 0

values

`-1` = Use HDR SYNC mode. Replication is synchronous if the primary server uses unbuffered logging.

`0` = The value of the HDR_TXN_SCOPE configuration parameter determines the synchronization mode for HDR data replication.

positive integers = Use HDR ASYNC mode. The positive integer is the maximum number of seconds between flushes of the data-replication buffer.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The DRINTERVAL configuration parameter controls replication latency, and is used to set the replication synchronization.

If used with unbuffered logging, HDR SYNC mode is the same as the nearly synchronous mode that is set through the HDR_TXN_SCOPE configuration parameter.

Table 66. Matrix of DRINTERVAL, HDR_TXN_SCOPE, and logging settings, and their resulting HDR replication modes.

DRINTERVAL	HDR_TXN_SCOPE	Logging	Result
-1	n/a	buffered	Asynchronous replication
-1	n/a	unbuffered	Nearly synchronous replication
0	FULL_SYNC	buffered	Fully synchronous replication
0	FULL_SYNC	unbuffered	Fully synchronous replication
0	ASYNC	buffered	Asynchronous replication
0	ASYNC	unbuffered	Asynchronous replication
0	NEAR_SYNC	buffered	Nearly synchronous replication

Table 66. Matrix of DRINTERVAL, HDR_TXN_SCOPE, and logging settings, and their resulting HDR replication modes. (continued)

DRINTERVAL	HDR_TXN_SCOPE	Logging	Result
0	NEAR_SYNC	unbuffered	Nearly synchronous replication
positive integer	n/a	buffered	Asynchronous replication
positive integer	n/a	unbuffered	Asynchronous replication

DRLOSTFOUND configuration parameter

Use the DRLOSTFOUND configuration parameter to specify the path name to the HDR lost-and-found file. This file indicates that some transactions were committed on the HDR primary database server before that were not committed on the secondary database server when the primary database server experienced a failure.

onconfig.std values

On UNIX™: `$ONEDB_HOME/etc/dr.lostfound`

On Windows™: `$ONEDB_HOME\tmp`

values

pathname = Path name of the `dr.lostfound` file

takes effect

After you edit your `onconfig` file and restart the database server.

The DRLOSTFOUND configuration parameter is not applicable if updates between the primary and secondary database servers occur synchronously, when the DRINTERVAL configuration parameter is set to `-1`.

The lost-and-found file, `dr.lostfound.`*`timestamp`*, is created with a time stamp that is appended to the file name so that the database server does not overwrite another lost and found file if another file exists. You cannot use the lost-and-found file to reapply lost transactions.

DRTIMEOUT configuration parameter

Use the DRTIMEOUT configuration parameter to specify the length of time, in seconds, that a database server in a high-availability data-replication pair waits for a transfer acknowledgment from the other database server in the pair. This parameter applies only to high-availability data-replication pairs.

onconfig.std value

DRTIMEOUT 30

values

Positive integers

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Use the following formula to calculate the value to specify for the DRTIMEOUT configuration parameter:

```
DRTIMEOUT = wait_time / 4
```

In this formula, *wait_time* is the length of time, in seconds, that a database server in a high-availability data-replication pair must wait before the server assumes that a high-availability data-replication failure occurred.

For example, you determine that *wait_time* for your system is 160 seconds. Use the preceding formula to set DRTIMEOUT as follows:

```
DRTIMEOUT = 160 seconds / 4 = 40 seconds
```

DS_HASHSIZE configuration parameter

Use the DS_HASHSIZE configuration parameter to specify the number of hash buckets in the data-distribution cache and other caches. The database server stores and accesses column statistics that the UPDATE STATISTICS statement generates in the MEDIUM or HIGH mode in the data-distribution cache.

onconfig.std value

DS_HASHSIZE 31

values

Any positive integer; a prime number is recommended

units

Number of hash buckets or lists

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Update the value of the DS_HASHSIZE and the DS_POOLSIZE configuration parameter to improve the performance of frequently used queries in a multiuser environment.

The DS_HASHSIZE configuration parameter sets the number of hash buckets for the following caches:

- Data-distribution cache
- Extend type name cache
- Extended type ID cache
- Cast cache
- Operator class instance cache
- Routine resolution cache
- Aggregate cache
- Secondary transient cache

DS_MAX_QUERIES configuration parameter

Use the DS_MAX_QUERIES configuration parameter to specify the maximum number of parallel database queries (PDQ) that can run concurrently.

The value of the DS_MAX_QUERIES configuration parameter is dependent on the setting for the DS_TOTAL_MEMORY configuration parameter:

- If the DS_TOTAL_MEMORY configuration parameter is set, then the value of the DS_MAX_QUERIES is `DS_TOTAL_MEMORY / 128`, rounded down to the nearest integer value.
- If the DS_TOTAL_MEMORY configuration parameter is not set, then the value of the DS_MAX_QUERIES configuration parameter is `2 * num`, where `num` is the number of CPUs specified in the VPCLASS configuration parameter.

onconfig.std value

Not set.

if not present

`2* num * 128`, where `num` is the number of CPUs specified in the VPCLASS configuration parameter.

values

Minimum value = 1

Maximum value = 8,388,608 (8 megabytes)

units

Number of queries

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The Memory Grant Manager (MGM) reserves memory for a query based on the following formula:

```
memory_reserved = DS_TOTAL_MEMORY *
                    (PDQ-priority / 100) *
                 (MAX_PDQPRIORITY / 100)
```

The value of PDQPRIORITY is specified in either the **PDQPRIORITY** environment variable or the SQL statement SET PDQPRIORITY.

DS_MAX_SCANS configuration parameter

Use the DS_MAX_SCANS configuration parameter to limit the number of PDQ scan threads that the database server can execute concurrently.

onconfig.std value

DS_MAX_SCANS 1048576 or (1024 * 1024)

values

10 - (1024 * 1024)

units

Number of PDQ scan threads

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

When a user issues a query, the database server apportions some number of scan threads, depending on the following values:

- The value of PDQ priority (set by the environment variable **PDQPRIORITY** or the SQL statement SET PDQPRIORITY)
- The ceiling that you set with DS_MAX_SCANS
- The factor that you set with MAX_PDQPRIORITY
- The number of fragments in the table to scan (*nfrags* in the formula)

The Memory Grant Manager (MGM) tries to reserve scan threads for a query according to the following formula:

```
reserved_threads = min (nfrags, (DS_MAX_SCANS *
                   PDQPRIORITY / 100 *
                  MAX_PDQPRIORITY / 100) )
```

If the DS_MAX_SCANS part of the formula is greater than or equal to the number of fragments in the table to scan, the query is held in the ready queue until as many scan threads are available as there are table fragments. Once underway, the query executes quickly because threads are scanning fragments in parallel.

For example, if *nfrags* equals `24`, DS_MAX_SCANS equals `90`, **PDQPRIORITY** equals `50`, and MAX_PDQPRIORITY equals `60`, the query does not begin execution until *nfrags* scan threads are available. Scanning takes place in parallel.

If the DS_MAX_SCANS formula falls below the number of fragments, the query might begin execution sooner, but the query takes longer to execute because some threads scan fragments serially.

If you reduce DS_MAX_SCANS to `40` in the previous example, the query needs fewer resources (12 scan threads) to begin execution, but each thread needs to scan two fragments serially. Execution takes longer.

DS_NONPDQ_QUERY_MEM configuration parameter

Use the DS_NONPDQ_QUERY_MEM configuration parameter to increase the amount of memory that is available for a query that is not a Parallel Database Query (PDQ). (You can only use this parameter if PDQ priority is set to zero.)

onconfig.std value

DS_NONPDQ_QUERY_MEM 128

DS_NONPDQ_QUERY_MEM:

On UNIX™: 256
On Windows™: 128

values

From the default value to 25 percent of the value of DS_TOTAL_MEMORY

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If you specify a value for the DS_NONPDQ_QUERY_MEM parameter, determine and adjust the value based on the number and size of table rows.

Tip: Set the value to generally not exceed the largest available temporary dbspace size.

The DS_NONPDQ_QUERY_MEM value is calculated during database server initialization based on the calculated DS_TOTAL_MEMORY value. If during the processing of the DS_NONPDQ_QUERY_MEM, the database server changes the value that you set, the server sends a message in this format:

```
DS_NONPDQ_QUERY_MEM recalculated and changed from old_value Kb to new_value Kb.
```

In the message, *old_value* represents the value that you assigned to DS_NONPDQ_QUERY_MEM in the user configuration file, and *new_value* represents the value determined by the database server.

DS_POOLSIZE configuration parameter

Use the DS_POOLSIZE parameter to specify the maximum number of entries in the data-distribution cache and other caches. The database server stores and accesses column statistics that the UPDATE STATISTICS statement generates in the MEDIUM or HIGH mode in the data-distribution cache.

onconfig.std value

DS_POOLSIZE 127

values

A positive value 127 or greater that represents the maximum number of entries in the cache. A positive value 127 or greater that represents half of the initial maximum number of entries in the cache. The maximum value is dependent upon the shared memory configuration and available shared memory for the server instance.

takes effect

After you edit your `onconfig` file and restart the database server.

When you increase the value in memory by running the onmode -wm command.

When you reset the value in memory by running the onmode -wm command.

Usage

Use the DS_HASHSIZE and the DS_POOLSIZE configuration parameters to improve performance of frequently run queries in a multi-user environment.

The initial number of entries in the cache is twice the value of the DS_POOLSIZE configuration parameter. For example, if the DS_POOLSIZE configuration parameter is set to 127, 254 entries are allowed in the cache. If all entries in a cache are full, the cache size automatically grows by 10%. To reduce the size of the cache, decrease the value of the DS_POOLSIZE configuration parameter in the `onconfig` file and restart the server.

The DS_POOLSIZE configuration parameter sets the number of entries in the following caches:

- Data-distribution cache
- Extend type name cache
- Extended type ID cache

- Cast cache
- Operator class instance cache
- Routine resolution cache
- Aggregate cache
- Secondary transient cache

DS_TOTAL_MEMORY configuration parameter

Use the DS_TOTAL_MEMORY configuration parameter to specify the amount of memory available for PDQ queries. The amount should be smaller than the computer physical memory, minus fixed overhead such as operating-system size and buffer-pool size.

onconfig.std value

Not set.

if not present

If SHMTOTAL=`0` and DS_MAX_QUERIES is set, `DS_TOTAL_MEMORY` = `DS_MAX_QUERIES * 128`.

If SHMTOTAL=`0` and DS_MAX_QUERIES is not set, `DS_TOTAL_MEMORY` = `num_cpu_vps * 2 * 128`.

values

If DS_MAX_QUERIES is set, the minimum value is `DS_MAX_QUERIES * 128`.

If DS_MAX_QUERIES is not set, the minimum value is `num_cpu_vps * 2 * 128`.

There is no maximum value limit other than any limit that you might have with the software that you use on your machine.

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Do not confuse DS_TOTAL_MEMORY with the configuration parameters SHMTOTAL and SHMVIRTSIZE. The SHMTOTAL setting specifies all the memory for the database server (total of the resident, virtual, and message portions of memory). The SHMVIRTSIZE setting specifies the size of the virtual portion. DS_TOTAL_MEMORY is a logical subset of SHMVIRTSIZE.

For OLTP applications, set DS_TOTAL_MEMORY to between 20 and 50 percent of the value of SHMTOTAL in kilobytes.

For applications that involve large decision-support (DSS) queries, increase the value of DS_TOTAL_MEMORY to between 50 and 80 percent of SHMTOTAL. If you use your database server for DSS queries exclusively, set this parameter to 90 and 100 percent of SHMTOTAL.

Set the DS_TOTAL_MEMORY configuration parameter to any value not greater than the quantity (`SHMVIRTSIZE - 10 megabytes`).

For information on the maximum memory available on your platform, see the machine notes.

Algorithm for DS_TOTAL_MEMORY

The database server derives a value for DS_TOTAL_MEMORY when you do not set DS_TOTAL_MEMORY, or if you set it to an inappropriate value. For information on the algorithms, see configuration effects on memory utilization in your *HCL OneDB™ Performance Guide*.

DUMPCNT configuration parameter (UNIX™)

Use the DUMPCNT configuration parameter to specify the number of assertion failures in a thread for which a database server dumps shared memory or generates a core file by calling the gcore utility.

onconfig.std value

DUMPCNT 1

values

Positive integers or a number of comma-separated fields as described in the Usage section.

units

Number of shared memory dumps or core files that can be generated by each thread.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

An assertion failure occurs when the database server cannot continue normal processing.

Assertion failures can generate as many core files or shared memory dumps as permitted by the DUMPCNT configuration parameter. Further assertion failures generate errors in the message log and perhaps to the application, but no further diagnostic information is saved.

```
>>-DUMPCNT--+----thread_count-----------------------+----><
            |                                       |
            |  '---------------------------,----'  |
            |  |                                |  |
            '--+-+-thrdlimit-=-thread_count----+-+--'
```

```
           |                             |
           +-instlimit-=-instance_count--+
           |                             |
           +-insttime-=-time_period------+
           |                             |
           '-interval-=-time_interval----'
```

Table 67. Options for the DUMPCNT configuration parameter value

Field	Values
thrdlimit	The *thread_count* value is a positive integer that represents the maximum number of shared memory dumps to be created by each thread. It also represents the maximum number of core files to be created by each thread if the DUMPGCORE onconfig parameter is enabled. The default is `1`.
instlimit	The *instance_count* value is a positive integer that represents the maximum number of shared memory dumps to be created by all threads in the database server instance in the time period specified by the insttime field. The default is `0` which implies no limit.
insttime	The *time_period* value is a positive integer to represent the period of time in seconds to be applied to the instlimit field. When the count of shared memory dumps reaches the limit set by instlimit then no further dumps will be created until the expiry of the time period. The start of the time period is the completion of the first shared memory dump. The default value is `0` which means that there is no time period and in this case, the instlimit field becomes an absolute count on the number of shared memory dumps to be created which remains in effect until the database server is restarted.
interval	The *time_interval* value is a positive integer representing the minimum time interval in seconds between the completion of one shared memory dump before another is permitted. The default value is `300 seconds`.

Note: When used, the fields instlimit, insttime, and interval are valid only for determination of creating a shared memory dump, not a core file. They are effective only when managed shared memory dumps are enabled by the DUMPSHMEM onconfig parameter. Furthermore, in order to prevent the loss of useful diagnostic information, they are not applied for an AFCRASH event.

DUMPCORE configuration parameter (UNIX™)

Use the DUMPCORE configuration parameter to control whether assertion failures cause a virtual processor to dump a core image. The core file is left in the directory from which the database server was last invoked. (The DUMPDIR parameter has no impact on the location of the core file.)

onconfig.std value

DUMPCORE 0

values

`0` = Do not dump core image.

`1` = Dump core image.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Warning: When DUMPCORE is set to 1, an assertion failure causes a virtual processor to dump a core image, which in turn causes the database server to abort. Set DUMPCORE only for debugging purposes in a controlled environment.

DUMPDIR configuration parameter

DUMPDIR specifies a directory in which the database server dumps shared memory, `gcore` files, or messages from a failed assertion.

Because shared memory can be large, set DUMPDIR to a file system with a significant amount of space. The directory to which DUMPDIR is set must exist for the server to start.

onconfig.std values

On UNIX™: `$ONEDB_HOME/tmp`

On Windows™: `$ONEDB_HOME\tmp`

values

Any directory to which user **informix** has write access

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

DUMPGCORE configuration parameter (UNIX™)

Use the DUMPGCORE configuration parameter to specify whether to dump the **gcore** core file. Use this configuration parameter with operating systems that support **gcore**.

onconfig.std value

DUMPGCORE 0

values

`0` = Do not dump **gcore**.

`1` = Dump **gcore**.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If you set DUMPGCORE, but your operating system does not support **gcore**, messages in the database server message log indicate that an attempt was made to dump a core image, but the database server cannot find the expected file. (If your operating system does not support **gcore**, set DUMPCORE instead.)

If DUMPGCORE is set, the database server calls **gcore** whenever a virtual processor encounters an assertion failure. The **gcore** utility directs the virtual processor to dump a core image to the `core.pid.cnt` file in the directory that DUMPDIR specifies and continue processing.

The **pid** value is the process identification number of the virtual processor. The **cnt** value is incremented each time that this process encounters an assertion failure. The **cnt** value can range from 1 to the value of DUMPCNT. After that, no more core files are created. If the virtual processor continues to encounter assertion failures, errors are reported to the message log (and perhaps to the application), but no further diagnostic information is saved.

DUMPSHMEM configuration parameter (UNIX™)

Use the DUMPSHMEM configuration parameter to indicate whether a shared memory dump is created on an assertion failure. This configuration parameter also specifies how much memory is written to the `shmem.`*`pid.cnt`* file in the directory specified by the DUMPDIR configuration parameter.

onconfig.std value

DUMPSHMEM 1

values

`0` = Do not create a shared memory dump.

`1` = Create a shared memory dump of all the shared memory that the database uses.

`2` = Create a shared memory dump that excludes the buffer pools.

`5` = Enables the managed shared memory dump feature. When permitted, create a shared memory dump of all the shared memory that the database uses.

`6` = Enables the managed shared memory dump feature. When permitted, create a shared memory dump that excludes the buffer pools.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If DUMPSHMEM is set to 1, all the shared memory that the database server uses is dumped, which can result in a large file. When space is limited, set DUMPSHMEM to 2 because this setting creates a smaller shared-memory dump file.

The values of 5 and 6 enables the managed shared memory dump feature. This determines whether or not a shared memory dump will be created depending on the type of Assertion Failure and how many shared memory dumps may have already been created by the thread or instance. When enabled the feature manages concurrent shared memory dump requests according to the type of Assertion Failure:

- A non-fatal request (AFWARN, AFFAIL) will be ignored if a shared memory dump is already in progress. The requesting thread will continue immediately.
- A fatal request (AFCRASH) will block the requesting thread if a shared memory dump is in progress. The thread is allowed to continue upon completion of the shared memory dump.

The DUMPCNT onconfig parameter provides options to control the working of the feature.

The *pid* value is the process identification number for the virtual processor. The *cnt* value increments each time that this virtual processor encounters an assertion failure. The *cnt* value can range from 1 to the value of the DUMPCNT configuration parameter. After the value of DUMPCNT is reached, no more files are created. If the database server continues to detect inconsistencies, errors are reported to the message log (and perhaps to the application), but no further diagnostic information is saved.

DYNAMIC_LOGS configuration parameter

Use the DYNAMIC_LOGS configuration parameter to allow logical logs to be dynamically added when necessary to prevent transaction blocking.

onconfig.std value

DYNAMIC_LOGS 2

values

`0` = Turn off dynamic-log allocation.

`1` = Set off the log file required alarm and pause to allow manual addition of a logical-log file. You can add a log file immediately after the current log file or to the end of the log file list.

`2` = Turn on dynamic-log allocation. When the database server dynamically adds a log file, it sets off the dynamically added log file alarm.

takes effect

For HDR: when the database server is shut down and restarted

For Enterprise Replication: when Enterprise Replication is started

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If DYNAMIC_LOGS is `2`, the database server automatically allocates a new log file when the next active log file contains an open transaction. Dynamic-log allocation prevents long transaction rollbacks from blocking transactions.

If you want to choose the size and location of the new logical-log file, set DYNAMIC_LOGS to `1`. Use the onparams -a command with the size (-s), location (-d dbspace), and -i options to add a log file after the current log file.

If the value of the DYNAMIC_LOGS configuration parameter is 0 and transaction blocking occurs, shut down the database server, set DYNAMIC_LOGS to `1` or `2`, and then restart the database server.

Important: If you are using Enterprise Replication with dynamic log allocation, set LTXEHWM to no higher than 70.

EILSEQ_COMPAT_MODE configuration parameter

Use the EILSEQ_COMPAT_MODE configuration parameter to control if HCL OneDB™ checks whether character data inserted by a client application contains code point sequences not recognized by the locale of the current database.

onconfig.std value

EILSEQ_COMPAT_MODE 0

values

`0` = HCL OneDB™ validates incoming character sequences with the current locale and returns error -202 if any characters are not valid.

`1` = HCL OneDB™ does not validate incoming character sequences.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

If you set the EILSEQ_COMPAT_MODE configuration parameter to `0`, only valid byte sequences can be inserted to the database.

The EILSEQ_COMPAT_MODE configuration parameter prevents a `202` error in these conditions:

- When data is being retrieved from the database.
- When an invalid character is at the end of the string and is a partial character.

ENABLE_SNAPSHOT_COPY configuration parameter

Use the ENABLE_SNAPSHOT_COPY configuration parameter to enable or disable the ability to clone a server using the ifxclone utility.

onconfig.std value

`0`

values

`0` = prohibit clone

`1` = permit clone

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The ENABLE_SNAPSHOT_COPY configuration parameter determines whether you can create a clone of a server using the ifxclone utility. Set the ENABLE_SNAPSHOT_COPY configuration parameter to 1 to allow cloning. Set the value to 0 to prohibit cloning the server using the ifxclone utility.

If you created a server during installation, the ENABLE_SNAPSHOT_COPY configuration parameter is enabled automatically.

ENCRYPT_CIPHERS configuration parameter

Use the ENCRYPT_CIPHERS configuration parameter to define all ciphers and modes that can be used by the current database session. ENCRYPT_CIPHERS is used for Enterprise Replication and High-Availability Data Replication only.

onconfig.std value

Not set. Encryption ciphers are not used.

values

See the Usage section.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The encryption cipher and mode used is randomly chosen among the ciphers common between the two servers. If a specific cipher is discovered to have a weakness, you should reset the ENCRYPT_CIPHERS configuration parameter value to eliminate that cipher by using the allbut option.

Important: Including all ciphers is more secure than including specific ciphers.

Syntax for the ENCRYPT_CIPHERS configuration parameter

`ENCRYPT_CIPHERS` { **`all`** | **`allbut`** `:<` { *cipher* | *mode* } `>` | *cipher*`:`*mode* }

Table 68. Options for the ENCRYPT_CIPHERS configuration parameter value

Field	Description
all	Include all available ciphers and modes, except ECB mode, which is considered weak. For example: `ENCRYPT_CIPHERS all`
allbut	Include all ciphers and modes, except ECB and the ciphers and modes listed. For example: `ENCRYPT_CIPHERS allbut:<cbc,bf>` The cipher list can include unique, abbreviated entries. For example, `bf` can represent bf-1, bf-2, and bf-3; however, if the abbreviation is the name of an actual cipher, then only that cipher is eliminated. Therefore, `des` eliminates only the des cipher, but `de` eliminates the des, des3, and desx ciphers.
cipher	The following ciphers are supported: • des = DES (64-bit key) • des3 = Triple DES • desx = Extended DES (128-bit key). Only supports cbc mode. • aes = AES 128bit key • aes192 = AES 192bit key • bf-1 = Blow Fish (64-bit key) • bf-2 = Blow Fish (128-bit key) • bf-3 = Blow Fish (192-bit key) • aes128 = AES 128bit key • aes256 = AES 256bit key

Table 68. Options for the ENCRYPT_CIPHERS configuration parameter value

(continued)

Field	Description
	All modes are supported for all ciphers, except the desx cipher. For an updated list of supported ciphers, see the Release Notes.
mode	The following modes are supported: • ecb = Electronic Code Book (ECB). Only included if specified. • cbc = Cipher Block Chaining • cfb = Cipher Feedback • ofb = Output Feedback

ENCRYPT_HDR configuration parameter

Use the ENCRYPT_HDR configuration parameter to enable or disable HDR encryption.

onconfig.std value

Not set.

values

`0` = Disables HDR encryption

`1` = Enables HDR encryption

takes effect

When the server is initialized

Usage

Enabling HDR encryption provides a secure method for transferring data from one server to another in an HDR pair. HDR encryption works in conjunction with Enterprise Replication (ER) encryption. However, it is not necessary to have ER encryption enabled for HDR encryption. HDR encryption works whether ER encryption is enabled or not. HDR and ER share the same encryption configuration parameters: ENCRYPT_CIPHERS, ENCRYPT_MAC, ENCRYPT_MACFILE and ENCRYPT_SWITCH.

ENCRYPT_MAC configuration parameter

Use the ENCRYPT_MAC configuration parameter to control the level of message authentication code (MAC) generation. This configuration parameter is used only for Enterprise Replication and High-Availability Data Replication.

onconfig.std value

Not set

values

`off` = Does not use MAC generation

`low` = Uses XOR folding on all messages

`medium` = Uses SHA1 MAC generation for all messages that are greater than 20 bytes long and XOR folding on smaller messages

`high` = Uses SHA1 MAC generation on all messages.

example

```
ENCRYPT_MAC medium,high
```

takes effect

For HDR: when the database server is shut down and restarted

For Enterprise Replication: when Enterprise Replication is started

Usage

The level is prioritized to the highest value. For example, if one node has a level of **high** and **medium** enabled and the other node has only **low** enabled, then the connection attempt fails. Use the **off** entry between servers only when a secure network connection is guaranteed.

ENCRYPT_MACFILE configuration parameter

Use the ENCRYPT_MACFILE configuration parameter to specify a list of the full path names of MAC key files. This configuration parameter is used only for Enterprise Replication and High-Availability Data Replication.

onconfig.std value

Not set.

values

One or more full path and file names separated by commas, and the optional **builtin** keyword. For example:

```
ENCRYPT_MACFILE /usr/local/bin/mac1.dat, /usr/local/bin/mac2.dat,builtin
```

units

Path names up to 1536 bytes in length

takes effect

For HDR: when the database server is shut down and restarted.

For Enterprise Replication: when Enterprise Replication is started.

Usage

Each of the entries for the ENCRYPT_MACFILE configuration parameter is prioritized and negotiated at connect time. The prioritization for the MAC key files is based on their creation time by the GenMacKey utility. The entry created from the **builtin** keyword has the lowest priority. Because the MAC key files are negotiated, you should periodically change the keys.

ENCRYPT_SMX configuration parameter

Use the ENCRYPT_SMX configuration parameter to set the level of encryption for high-availability configurations on secondary servers and between Enterprise Replication Servers.

Note: From version 2.0.0.0 onwards, Enterprise Replication will use SMX connection for communicating with the peer servers.

onconfig.std value

Not set.

values

`0` = Off. Do not encrypt.

`1` = On. Encrypt where possible. Encrypt SMX transactions when the database server being connected to also supports encryption.

`2` = On. Always encrypt. Only connections to encrypted database servers are allowed.

takes effect

After you edit your `onconfig` file and restart the database server.

ENCRYPT_SWITCH configuration parameter

Use the ENCRYPT_SWITCH configuration parameter to define the frequency at which ciphers or secret keys are renegotiated. This configuration parameter is used only for Enterprise Replication and High-Availability Data Replication.

The longer the secret key and encryption cipher remains in use, the more likely the encryption rules might be broken by an attacker. To avoid this, cryptologists recommend changing the secret keys on long-term connections. The default time that this renegotiation occurs is once an hour.

onconfig.std value

Not set.

values

Two positive integers separated by a comma. The first integer represents the number of minutes between cipher renegotiation. The second integer represents the number of minutes between secret key renegotiation. For example: ENCRYPT_SWITCH 2,5.

units

minutes

takes effect

For HDR: when the database server is shut down and restarted

For Enterprise Replication: when Enterprise Replication is started

EXPLAIN_STAT configuration parameter

Use the EXPLAIN_STAT configuration parameter to enable or disable the inclusion of a Query Statistics section in the explain output file.

You can generate the output file by using either the SET EXPLAIN statement or the onmode -Y *sessionid* command. When you enable the EXPLAIN_STAT configuration parameter, the Query Statistics section shows the estimated number of rows and the actual number of returned rows in the Query Plan.

onconfig.std value

EXPLAIN_STAT 1

values

`0` = Disable the inclusion of a Query Statistics section in the explain output file.

`1` = Enable the inclusion of a Query Statistics section in the explain output file.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

EXT_DIRECTIVES configuration parameter

Use the EXT_DIRECTIVES configuration parameter to enable or disable the use of external query optimizer directives.

onconfig.std value

EXT_DIRECTIVES 0

values

`0` (default) = Off. The directive cannot be enabled even if **IFX_EXTDIRECTIVES** is on.

`1` = On. The directive can be enabled for a session if **IFX_EXTDIRECTIVES** is on.

`2` = On. The directive can be used even if **IFX_EXTDIRECTIVES** is not set.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Enable external directives by using the EXT_DIRECTIVES configuration parameter in combination with the client-side **IFX_EXTDIRECTIVES** environment variable as follows:

The setting of the **IFX_EXTDIRECTIVES** environment variable overrides the setting of the EXT_DIRECTIVES configuration parameter. If you do not set the **IFX_EXTDIRECTIVES** environment variable, all sessions for a client inherit the database server configuration for processing external directives.

The setting specified by the SET ENVIRONMENT EXTDIRECTIVES statement of SQL overrides (for the current user session only) the settings of both the **IFX_EXTDIRECTIVES** environment variable and of the EXT_DIRECTIVES configuration parameter.

EXTSHMADD configuration parameter

Use the EXTSHMADD configuration parameter to specify the size of virtual-extension segments that are added when a user-defined routine or a DataBlade® routine is run in a user-defined virtual processor.

onconfig.std value

EXTSHMADD 8192

values

32-bit operating systems: 1024 - 524288

64-bit operating systems: 1024 - 4294967296

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

When a thread is run in a user defined virtual processor, a virtual-extension segment is created. In the output of the onstat -g seg command, the virtual-extension segment has a class of VX. If the EXTSHMADD configuration parameter is not set in the `onconfig` file, the size of virtual-extension segments is set by the value of the SHMADD configuration parameter.

FAILOVER_CALLBACK configuration parameter

Use the FAILOVER_CALLBACK configuration parameter to specify the script executed by the database server when a database server transitions from a secondary server to a primary or standard server.

onconfig.std value

Not set.

values

pathname = The full path name of the script specified by the FAILOVER_CALLBACK parameter.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Set FAILOVER_CALLBACK to the full path name of the script.

FAILOVER_TX_TIMEOUT configuration parameter

In high-availability cluster environments, use the FAILOVER_TX_TIMEOUT configuration parameter to enable transactions to complete after failover of the primary server.

Use the FAILOVER_TX_TIMEOUT configuration parameter to indicate the maximum number of seconds after failover that the server waits before it begins rolling back transactions. Set the FAILOVER_TX_TIMEOUT configuration parameter to the same value on all servers in a high-availability cluster.

onconfig.std value

FAILOVER_TX_TIMEOUT 0

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

When a failover occurs in a high-availability cluster environment, one of the secondary servers takes over the role of the primary server. The secondary server that becomes the new primary server is called the *failover server*.

You enable transaction survival by setting the FAILOVER_TX_TIMEOUT configuration parameter to a value greater than zero. When transaction survival is enabled, the failover server must be able to contact the remaining secondary servers to synchronize and resume any open transactions. Similarly, the surviving secondary servers must be able to establish connections to the failover server to re-send any pending transactions. The FAILOVER_TX_TIMEOUT configuration parameter specifies how long the servers wait before they begin rolling back transactions.

On the failover server, if the number of seconds specified by FAILOVER_TX_TIMEOUT is exceeded, any open transactions that are not synchronized with a surviving server are terminated and rolled back.

On the remaining secondary servers, if the number of seconds specified by FAILOVER_TX_TIMEOUT is exceeded, any open transactions on that server return an error.

Set FAILOVER_TX_TIMEOUT to 0 to immediately roll back all open transactions when failover occurs.

If the primary server fails and a secondary server fails to take over the role of the primary server, then any open transactions are rolled back, and the client is unable to make updates. For example, if an update activity has been started on a secondary server and the primary server fails, and then that failover processing does not complete and a new primary server is not established, after a predetermined amount of time, the client request times out, placing the sqlexec thread in an indeterminate state.

In the preceding scenario, active transactions are rolled back, but the physical rollback cannot occur until the new primary server is established (because the primary server manages the logs). Under these circumstances, the session can be unaware of operations that were performed on the secondary server. The session can be unaware of the rollback of a partially applied transaction because the rollback of the partial transaction cannot occur until a new primary server is established.

FASTPOLL configuration parameter

Use the FASTPOLL configuration parameter to enable or disable fast polling of your network. FASTPOLL is a platform-specific configuration parameter.

onconfig.std value

FASTPOLL 1

values

`0` = Disables fast polling.

`1` = Enables fast polling.

takes effect

After you edit your `onconfig` file and restart the database server.

FILLFACTOR configuration parameter

Use the FILLFACTOR configuration parameter to specify the degree of index-page fullness. A low value provides room for growth in the index. A high value compacts the index.

If an index is full (100 percent), any new inserts result in splitting nodes. You can also set the FILLFACTOR as an option on the CREATE INDEX statement. The setting on the CREATE INDEX statement overrides the ONCONFIG file value.

You cannot use the FILLFACTOR configuration parameter with a forest of trees index.

onconfig.std value

FILLFACTOR 90

values

1 - 100

units

Percent

takes effect

When the index is built. Existing indexes are not changed. To use the new value, the indexes must be rebuilt.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

FULL_DISK_INIT configuration parameter

Use the FULL_DISK_INIT configuration parameter to prevent an accidental disk reinitialization of an existing database server instance. This configuration parameter specifies whether or not the disk initialization command (oninit -i) can run on your HCL OneDB™ instance when a page zero exists at the root path location, which is at the first page of the first chunk location.

onconfig.std value

FULL_DISK_INIT 0

values

`0` = The oninit -i command runs only if there is not a page zero at the root path location.

`1` = The oninit -i command runs under all circumstances, but also resets the FULL_DISK_INIT configuration parameter to `0` after the disk initialization.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

When the FULL_DISK_INIT configuration parameter is set to `1`, any instance startup command (for example, oninit as well as oninit -i) resets the configuration parameter to `0`.

If you start to run the oninit -i command when the FULL_DISK_INIT configuration parameter is set to `0` and the database server finds a page zero, the oninit -i command does not run and the server reports an error in the `online.log`.

Page zero is the HCL OneDB™ system page that contains general information about the server instance. This page is created when the server instance is initialized.

HA_ALIAS configuration parameter

The HA_ALIAS configuration parameter defines a network alias that is used for server-to-server communication in a high-availability cluster. The specified network alias is also used by Connection Managers, the ifxclone utility, and onmode -d commands.

onconfig.std value

Not set. The HA_ALIAS configure parameter applies to high-availability cluster servers.

values

The HA_ALIAS configuration parameter value must match a DBSERVERNAME or DBSERVERALIASES configuration parameter value that is associated with a TCP `sqlhosts` file entry. If the DBSERVERNAME or the DBSERVERALIASES configuration parameter value includes the optional number of listener threads, omit the optional listener thread value from the HA_ALIAS configuration parameter value. For example, if DBSERVERNAME is set to `my_server-4`, HA_ALIAS is set to `my_server`.

takes effect

After you edit your `onconfig` file and restart the database server.

For the primary server in a high-availability cluster, reset the value dynamically in your `onconfig` file by running the onmode -wf command. This method does not work for secondary servers in a high-availability cluster.

For the primary server in a high-availability cluster, reset the value in memory by running the onmode -wm command. This method does not work for secondary servers in a high-availability cluster.

Usage

The HA_ALIAS configuration parameter is required for high-availability cluster servers that use shared-memory connections.

For example, if a high-availability cluster server's DBSERVERNAME configuration parameter is associated with a shared-memory `sqlhosts` file entry, set a DBSERVERALIAS configuration parameter and a matching HA_ALIAS configuration parameter value, and then create a TCP `sqlhosts` file entry for the for the alias.

`onconfig` file values:

```
DBSERVERNAME my_server
DBSERVERALIAS alias_1
HA_ALIAS alias_1
```

`sqlhosts` file values:

```
#dbservername  nettype   hostname  servicename  options
 my_server     onipcshm  host_1    port_1              #client-to-server
 alias_1       onsoctcp  host_1    port_2              #server-to-server
```

Setting the HA_ALIAS configuration parameter for all servers in a high-availability cluster also enables you to separate client/server communication from server-to-server communication .

HA_FOC_ORDER configuration parameter

Use the HA_FOC_ORDER configuration parameter to define a single connection-management failover rule for a high-availability cluster of servers.

onconfig.std value

HA_FOC_ORDER SDS,HDR,RSS

values

A list of secondary server types, which are separated by commas and listed in priority order. For example, the default value of `SDS,HDR,RSS` means that the primary server fails over to the SD secondary server, then the HDR secondary server, and then the RS secondary server.

- `HDR` = High-availability data replication server
- `RSS` = Remote stand-alone secondary server
- `SDS` = Shared-disk secondary server

`MANUAL` = Disable automated failover for all Connection Managers in the cluster.

separators

Separate values with a comma.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

After you run the SQL administration API task() or admin() function with the -wf HA_FOC_ORDER=*value* or -wm HA_FOC_ORDER=*value* arguments.

Usage

If the HA_FOC_ORDER configuration parameter is set on the primary database server of a high-availability cluster, every Connection Manager that connects to the primary server adopts the setting. The value replaces the connection unit's `ORDER=rule` failover-sequence rule. Each database server in the high-availability cluster then adopts the primary server's HA_FOC_ORDER configuration parameter value for its own HA_FOC_ORDER configuration parameter.

If the HA_FOC_ORDER configuration parameter on the primary server is set to `MANUAL`, automated failover is disabled on all Connection Managers that manager the primary server's cluster.

If the FOC ORDER value for a connection unit in a Connection Manager's configuration file is set to `DISABLED` the Connection Manager does not perform failover for that connection unit.

Syntax for the HA_FOC_ORDER configuration parameter

HA_FOC_ORDER { { SDS | HDR | RSS } | MANUAL }

Example

Example

In the following example, you have two Connection Managers that are configured to manage a cluster of three servers.

The three servers are:

- **server_1** (primary server)
- **server_2** (SD secondary server)
- **server_3** (HDR secondary server)

The first Connection Manager has the following configuration file:

```
NAME connection_manger_1

CLUSTER cluster_1
{
   ONEDB_SERVER servers_1
   SLA sla_1 DBSERVERS=ANY
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

The second Connection Manager has the following configuration file:

```
NAME connection_manger_2

CLUSTER cluster_1
{
   ONEDB_SERVER servers_1
   SLA sla_2 DBSERVERS=ANY
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

The `onconfig` file of **server_1** has the following value:

```
HA_FOC_ORDER SDS,HDR
```

When **connection_manger_1** and **connection_manger_2** connect with **server_1**, their configurations become:

```
NAME connection_manger_1

CLUSTER cluster_1
{
   ONEDB_SERVER servers_1
   SLA sla_1 DBSERVERS=ANY
   FOC ORDER=SDS,HDR \
       PRIORITY=1
}
```

```
NAME connection_manger_2

CLUSTER cluster_1
{
   ONEDB_SERVER servers_1
   SLA sla_2 DBSERVERS=ANY
   FOC ORDER=SDS,HDR \
       PRIORITY=2
}
```

The values of the HA_FOC_ORDER entries in the `onconfig` files of **server_2** and **server_3** are updated to `SDS,HDR`.

HDR_TXN_SCOPE configuration parameter

The HDR_TXN_SCOPE configuration parameter is used with the DRINTERVAL configuration parameter to specify the synchronization mode for HDR replication in a high-availability cluster.

onconfig.std value

HDR_TXN_SCOPE NEAR_SYNC

values

FULL_SYNC = HDR replication if fully synchronous. Transactions require acknowledgement of completion on the HDR secondary server before they can complete.

NEAR_SYNC = HDR replication if nearly synchronous. Transactions require acknowledgement of being received on the HDR secondary server before they can complete. If used with unbuffered logging, SYNC mode, which is turned on when DRINTERVAL is set to -1, is the same as nearly synchronous mode.

ASYNC = HDR replication if fully asynchronous. Transactions do not require acknowledgement of being received or completed on the HDR secondary server before they can complete.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

After you run the SQL administration API task() or admin() function with the `"onmode","-wf HDR_TXN_SCOPE=value"` or `"onmode","-wm HDR_TXN_SCOPE=value"` argument.

Usage

When the DRINTERVAL configuration parameter is set to 0, the value of the HDR_TXN_SCOPE parameter determines the synchronization mode for HDR replication.

If unbuffered logging is used, HDR SYNC mode is the same as the nearly synchronous mode that is set through the HDR_TXN_SCOPE configuration parameter.

Table 69. Matrix of DRINTERVAL, HDR_TXN_SCOPE, and logging settings, and their resulting HDR replication modes.

DRINTERVAL	HDR_TXN_SCOPE	Logging	Result
-1	n/a	buffered	Asynchronous replication
-1	n/a	unbuffered	Nearly synchronous replication
0	FULL_SYNC	buffered	Fully synchronous replication

Table 69. Matrix of DRINTERVAL, HDR_TXN_SCOPE, and logging settings, and their resulting HDR replication modes. (continued)

DRINTERVAL	HDR_TXN_SCOPE	Logging	Result
0	FULL_SYNC	unbuffered	Fully synchronous replication
0	ASYNC	buffered	Asynchronous replication
0	ASYNC	unbuffered	Asynchronous replication
0	NEAR_SYNC	buffered	Nearly synchronous replication
0	NEAR_SYNC	unbuffered	Nearly synchronous replication
positive integer	n/a	buffered	Asynchronous replication
positive integer	n/a	unbuffered	Asynchronous replication

IFX_EXTEND_ROLE configuration parameter

Your database system administrator (DBSA), by default user **informix**, can use the IFX_EXTEND_ROLE parameter to control which users are authorized to register DataBlade® modules or external user-defined routines (UDRs).

onconfig.std value

IFX_EXTEND_ROLE 1

values

`1` or `On` (default) = Enables the requirement for the EXTEND role so that administrators can grant privileges to a user to create or drop a UDR that includes the EXTERNAL clause.

`0` or `Off` = Disables the requirement for the EXTEND role, so that any user who holds the USAGE ON LANGUAGE privilege for the appropriate external language (C or JAVA) can register or drop an external routine that was written in that language.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

IFX_FOLDVIEW configuration parameter

Use the IFX_FOLDVIEW configuration parameter to enable or disable view folding. For certain situations where a view is involved in a query, view folding can significantly improve the performance of the query. In these cases, views are folded into a parent query instead of the query results being put into a temporary table.

onconfig.std value

IFX_FOLDVIEW 1

values

`0` or `Off` = Disables view folding.

`1` or `On` = Default. Enables view folding.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The following types of queries can take advantage of view folding:

- Views that contain a UNION ALL clause and the parent query includes a regular join, HCL OneDB™ join, ANSI join, or an ORDER BY clause

A temporary table is created and view folding is not performed for the following types of queries that perform a UNION ALL operation involving a view:

- The view has one of the following clauses: AGGREGATE, GROUP BY, ORDER BY, UNION, DISTINCT, or OUTER JOIN (either HCL OneDB™ or ANSI type).
- The parent query has a UNION or UNION ALL clause.

IFX_XA_UNIQUEXID_IN_DATABASE configuration parameter

Use the IFX_XA_UNIQUEXID_IN_DATABASE configuration parameter to enable the transaction manager to use the same XID to represent global transactions on different databases in the same database server instance.

onconfig.std value

None

default value

0

values

`0` = disabled

`1` = enabled

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

An XID is a global transaction ID for a distributed XA transaction.

If you set the IFX_XA_UNIQUEXID_IN_DATABASE configuration parameter to 1, the database server allows the transaction manager to use the same XID to represent global transactions on different databases in the same database server instance. Thus, the database can be the domain instead of the server.

CONNECT_RETRIES configuration parameter

Use the **CONNECT_RETRIES** configuration parameter to specify the maximum number of connection attempts that can be made to each database server after the initial connection attempt fails. These attempts are made within the time limit that the **CONNECT_TIMEOUT** configuration parameter specifies.

onconfig.std value

CONNECT_RETRIES 1

values

Positive integers

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The **CONNECT_TIMEOUT** setting takes precedence over the **CONNECT_RETRIES** setting. Connection attempts can end after the **CONNECT_TIMEOUT** value is exceeded, but before the **CONNECT_RETRIES** value is reached.

To override the value of the **CONNECT_RETRIES** configuration parameter for the current session, you can set either the **CONNECT_RETRIES** environment option of the SET ENVIRONMENT statement or the client's **CONNECT_RETRIES** environment variable.

CONNECT_TIMEOUT configuration parameter

Use the **CONNECT_TIMEOUT** configuration parameter to specify the number of seconds that the CONNECT statement attempts to establish a connection to a database server.

onconfig.std value

CONNECT_TIMEOUT 60

values

Positive integers

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

To set the optimal value for the **CONNECT_TIMEOUT** configuration parameter, take into account the total distance between nodes, the hardware speed, the volume of traffic, and the concurrency level of the network.

The **CONNECT_TIMEOUT** value is divided by the **CONNECT_RETRIES** value to determine the number of seconds between connection attempts. If you set the **CONNECT_TIMEOUT** configuration parameter to zero, the database server uses the default value of 60 seconds.

To override the value of the **CONNECT_TIMEOUT** configuration parameter for the current session, you can set either the **CONNECT_TIMEOUT** environment option of the SET ENVIRONMENT statement or the client's **CONNECT_TIMEOUT** environment variable.

LICENSE_SERVER configuration parameter

The LICENSE_SERVER configuration parameter specifies the OneDB instance about the FlexNet Server Device it needs to connect to obtain the necessary CPU license resources.

onconfig.std value

Not set.

values

For the cloud based license server, a URL that points to the FlexNet Operations license server API, with **<Device_ID>** set to the Server Device name.

https://hclsoftware.compliance.flexnetoperations.com/instances/**<Device_ID>**/ request

For a local license server, the server will be the hostname and port number of the machine running the LLS.

https://corporate-lls.local:7070/instances/**<Device_ID>**/request

takes effect

On starting the instance, the FlexNet server indicated is queried to determine if sufficient licenses are available.

Usage

The HCL OneDB Server instance must obtain sufficient license resources to start up all the CPU VPs specified in the onconfig file. It will fail to start if the licenses are not available. Any subsequent dynamic change to the CPU VP configuration will require rechecking the license entitlement with the FlexNet license server. The OneDB instance will disable CPU VPs if there are no available licenses.

Care should be taken with the VPCLASS and AUTO_TUNE parameters to ensure that the correct initial request for CPU resource is made, and that any CPU VPs dynamically added by the system have sufficient licenses and do not cause license starvation issues for other instances using the same license server device.

If there are possible issues with license allocation between multiple OneDB instances, using the same FlexNet Server Device, then it is possible to tie the license allocations to specific instances using FlexNet's Reservation Group feature.

License allocations are returned when the OneDB instance is stopped. If an instance fails, and is unable to return the licenses, then the licenses will be re-allocated when the server is restarted. If it is not possible to restart the server after a failure, and the licenses need to be returned for use by another instance, then the onclean utility can be used to return the licenses.

A Local License server cannot be installed on a virtual machine. The OneDB instance, acting as a Client Device, accesses the license server through HTTP to a nominated port number, the default being 7070. Control and monitoring of the LLS through the REST interface is accomplished through a separate HTTPS port.

LICENSE_TIMING configuration parameter

Use the LICENSE_TIMING configuration parameter to configure the interval that OneDB instance will borrow FlexNet licenses for CPU license resources, and the interval that the instance will renew the license.

onconfig.std value

LICENSE_TIMING 28days,27days

values

Range for borrow_interval between 2hours and 30days. Range for renewal_interval between 1hour and 719hours (29days is max if specifying in day units).

separators

Separate values with a comma.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

After you run the SQL administration API **task()** or **admin()** function with the **-wf** LICENSE_TIMING **=***value* or **-wm**LICENSE_TIMING **=***value* arguments.

Usage

Use this parameter to set the how long the instance wants to use the license (borrow_interval), and how often the instance should renew the license (renewal_interval). An interval period can be specified in days, or hours, using the format Ndays or Nhours where N is a number greater or equal to 0. The use of day, d, hour or h are acceptable short forms.

A value of “2days,25hours? means that the instance will request to borrow licenses for 2 days, with a minimum renewal duration of 25 hours. So every 23 hours we will contact the license server and renew the licenses assigned to the current device. There is also an hourly check to see that the license server is still available. If the license server cannot be contacted, warning messages will sent to the log and an alarm raised through ALRAMPROGRAM.

A value “28days,27days? means that the instance will request to borrow licenses for 28 days, with a minimum renewal duration of 27 days. So every 24 hours we will contact the license server and renew the licenses assigned to the current device.

LIMITNUMSESSIONS configuration parameter

Use the LIMITNUMSESSIONS configuration parameter to define the maximum number of sessions that you want connected to HCL OneDB™.

If you specify a maximum number, you can also specify whether you want HCL OneDB™ to print messages to the `online.log` file when the number of sessions approaches the maximum number.

If the LIMITNUMSESSIONS configuration parameter is enabled and sessions are restricted because of this limit, both regular user threads and DBSA user threads connecting to any database count against the limit. However, a DBSA user is allowed to connect to the server even after the limit has been reached.

Distributed queries against a server are also counted against the limit.

The LIMITNUMSESSIONS configuration parameter is not intended to be used as a means to adhere to license agreements.

onconfig.std value

Not set in the `onconfig.std` file

values

maximum_number_of_sessions = `0` to 2,097,152 (2*1024*1024). The default is `0`.

print_warning = `0` (off) or `1` (on). The default for this optional value is `0`.

separators

Comma

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If the print_warning is set to `1`, a warning is triggered when the number of sessions is greater than or equal to 95 percent of the maximum_number_of_sessions value. If print_warning is set to zero, or if it is not set, no warning is issued No new user sessions can be opened after the maximum_number_of_sessions limit is reached.

If the maximum_number_of_sessions value for the LIMITNUMSESSIONS configuration parameter is set to `0`, or if it is not set, there is no limit to the number of sessions that can connect to the server.

The following example specifies that you want a maximum of 100 sessions to connect to the server and you want to print a warning message when the number of connected sessions approaches 100.

```
LIMITNUMSESSIONS 100,1
```

The settings in this example cause a warning to be printed when more than 94 sessions are concurrently connected. Only a member of the DBSA group can start a new session when 100 sessions are already connected.

Use onmode -wf or onmode -wm, or the equivalent SQL administration API ONMODE commands, to dynamically increase or temporarily disable the LIMITNUMSESSIONS setting. Use this configuration parameter to allow administrative utilities to run if the database server is reaching the maximum_number_of_sessions limit.

LISTEN_TIMEOUT configuration parameter

Use the LISTEN_TIMEOUT configuration parameter to specify the number of seconds in which the server waits for a connection.

onconfig.std value

LISTEN_TIMEOUT 60

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

You can set LISTEN_TIMEOUT to a lower number to guard against faulty connection requests that might indicate a Denial of Service attack.

Depending on the machine capability of holding the threads (in number), you can configure MAX_INCOMPLETE_CONNECTIONS to a higher value and depending on the network traffic, you can set LISTEN_TIMEOUT to a lower value to reduce the chance that an attack can reach the maximum limit.

LOCKS configuration parameter

The LOCKS configuration parameter specifies the initial size of the lock table.

The lock table holds an entry for each lock. If the number of locks allocated exceeds the value of the LOCKS configuration parameter, the database server increases the size of the lock table. The lock table can be increased a maximum of 99 times.

onconfig.std value

LOCKS 20000

values

2,000 through 8,000,000 for 32-bit database servers 2,000 through 500,000,000 for 64-bit database servers

units

Number of locks in the internal lock table

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The database server increases the size of the lock table by attempting to double the lock table on each increase. However, the amount added during each increase is limited to a maximum value. For 32-bit platforms, a maximum of 100,000 locks can be added during each increase. Therefore, the total maximum locks allowed for 32-bit platforms is 8,000,000 (maximum number of starting locks) + (99 (maximum number of dynamic lock table extensions) x 100,000 (maximum number of locks added per lock table extension). For 64-bit platforms, a maximum of 1,000,000 locks can be added during each increase. Therefore, the total maximum locks allowed is 500,000,000 (maximum number of starting locks) + (99 (maximum number of dynamic lock table extensions) x 1,000,000 (maximum number of locks added per lock table extension).

With the initial lock table stored in resident memory and each additional lock stored in virtual memory, locks can become a resource drain if you have a limited amount of shared memory. The amount of storage occupied by a single lock depends on the word size and operating system, and is subject to change. Currently, the amount of storage ranges from approximately 100 to 200 bytes. You can see the amount of storage required to support additional locks by restarting the server with a different value of the LOCKS configuration parameter (without making other changes), and observing the increase in memory used as shown by "onstat -g mem" for the resident pool.

Tip: When you drop a database, a lock is acquired and held on each table in the database until the database is dropped.

LOGBUFF configuration parameter

Use the LOGBUFF configuration parameter to specify the size in kilobytes for the three logical-log buffers in shared memory.

onconfig.std value

LOGBUFF 64

units

Kilobytes

values

An integer in the range of 32 - (32767 * *pagesize* / 1024), where *pagesize* is the default system page size. The value must be evenly divisible by the default system page size. If the value is not evenly divisible by the page size, the database server rounds down the size to the nearest value that is evenly divisible by the page size.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The three logical log buffers permit user threads to write to the active buffer while one of the other buffers is being flushed to disk. If flushing is not complete by the time the active buffer fills, the user thread begins writing to the third buffer.

If the RTO_SERVER_RESTART configuration parameter is enabled, set the value of the LOGBUFF configuration parameter to 256 kilobytes. If the value of the LOGBUFF configuration parameter is less than 256 kilobytes, a warning message displays when you restart the server.

Otherwise, set the value of the LOGBUFF configuration parameter to 32 kilobytes for standard workloads or 64 kilobytes for heavy workloads. The database server uses the LOGBUFF parameter to set the size of internal buffers that are used during recovery. If you set LOGBUFF too high, the database server can run out of memory and shut down during recovery.

If you log user data in smart large objects, increase the size of the log buffer to make the system more efficient. The database server logs only the portion of a smart-large-object page that changed.

You can view information about the logical log buffers by running the onstat -l command.

LOGBUF_INTVL configuration parameter

Use the LOGBUF_INTVL configuration parameter to ensure the logical log buffer is flushed periodically when only buffered logging is used.

onconfig.std value

LOGBUF_INTVL 0

units

Seconds

values

The integer value of the parameter is the maximum number of seconds between logical log buffer flushes.

Default value: 0 – The feature is disabled by default. The time since the last logical log buffer flush will not be used as a criterion for flushing the buffer.

Minimum value = `1`

Maximum value = `2147483647`

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Setting this parameter will also positively affect latency in a replication environment when buffered logging is used. During relatively quiet periods on the primary, log records may be pushed to secondaries more frequently.

LOGFILES configuration parameter

Use the LOGFILES configuration parameter to specify the number of logical-log files that the database server creates during disk initialization.

onconfig.std value

LOGFILES 6

values

3 - 32,767 (integers only)

units

Number of logical-log files

takes effect

During disk initialization and when you add a new log file. You add a new log with one of the onparms utilities.

Usage

To change the number of logical-log files, add or drop logical-log files.

If you use onparams to add or drop log files, the database server automatically updates LOGFILES.

If you use ISA or onparams to add or drop log files, the database server automatically updates LOGFILES.

LOG_INDEX_BUILDS configuration parameter

Use the LOG_INDEX_BUILDS configuration parameter to enable or disable index page logging.

onconfig.std value

Not set.

values

`0` = Disable

`1` = Enable

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If LOG_INDEX_BUILDS is enabled, logical log file space consumption will increase, depending on the size of the indexes. This might lead to logical log file backups being required more frequently. Messages are written to the `online.log` file when index page logging status changes.

Tip for RS secondary servers: Using onmode -wm enables or disables index page logging for the current session only, and does not affect the setting in the onconfig file. If the server is stopped and restarted, the setting in the onconfig file determines whether index page logging is enabled. Therefore, enabling index page logging using onmodem -wm is not recommended when using RS secondary servers; instead, use onmode -wf to update the onconfig file, so that index page logging is enabled after restarting the server. Index page logging is a requirement when using RS secondary servers.

LOG_STAGING_DIR configuration parameter

Use the LOG_STAGING_DIR configuration parameter to specify the location of log files received from the primary server when configuring delayed application of log files on RS secondary servers.

onconfig.std value

Not set.

values (first parameter)

Any valid, secure directory.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The LOG_STAGING_DIR configuration parameter specifies the directory where log files sent from the primary are stored in the following circumstances:

- The DELAY_APPLY configuration parameter is set on an RS secondary server to delay the application of logs
- The STOP_APPLY configuration parameter is set on an RS secondary server to stop the application of logs
- An RS secondary server must temporarily buffer logs
- The LOG_INDEX_BUILDS parameter is set on the HDR secondary server, and the HDR secondary server is processing checkpoints

Delaying the application of log files allows you to recover quickly from erroneous database modifications by restoring the data from the RS secondary server.

The directory specified by the LOG_STAGING_DIR configuration parameter must be secure. The directory must be owned by user **informix**, must belong to group **informix**, and must not have public read, write, or execute permission.

The directory should have enough space to hold all the logical logs that are staged. Choose a directory capable of storing at least twice the total logical logs on the primary server. To estimate the storage size, multiply the value of the LOGBUFF configuration parameter with the value of the LOGFILES configuration parameter, and then double that value.

To see information about the data being sent to the log-staging directory set for a RS secondary server, run the onstat -g rss verbose command on the RS secondary server.

If the write to the staging file fails, the RS secondary server raises event alarm 40007.

LOGSIZE configuration parameter

Use the LOGSIZE configuration parameter to specify the size that is used when logical-log files are created.

onconfig.std value

LOGSIZE 10000

units

Kilobytes

values

An integer value.

Minimum value = `200`

Maximum value when the database server is first initialized = `(ROOTSIZE - PHYSFILE - 512 - (63 * pagesize/1024)) / LOGFILES`

The *pagesize* value is the default system page size for the operating system.

If you expand the root dbspace or move logical logs to a different dbspace, the maximum size of logical log files cannot exceed the following page size-dependent value:

- 1 GiB for page size = 2 KiB
- 2 GiB for page size = 4 KiB

This limit is the maximum number of pages that the log position can describe for those page sizes.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

When you change the value of the LOGSIZE configuration parameter, only new log files are affected. The size of existing log files does not change. The total logical-log size is the product of the LOGSIZE configuration parameter setting multiplied by the value of the LOGFILES configuration parameter. However, if you change the value of the LOGSIZE configuration parameter, the total size of all logical log files depends on the number of log files of each size.

If the AUTO_LLOG configuration parameter is enabled, logical log files are added automatically as needed to improve performance, up to a configurable maximum total logical-log size.

To verify the page size that the database server uses on your platform, run the onstat -b command.

If you declare logging for a smart-large-object column, you must ensure that the logical log is considerably larger than the amount of data that is logged during inserts or updates. The database server cannot back up open transactions. If many transactions are active, the total logging activity must not force open transactions to the log backup files. For example, if your log size is 1000 KB and the high-watermark is 60 percent, do not use more than 600 KB of the logical log for the smart-large-object updates. The database server starts rolling back the transaction when it reaches the high-watermark of 600 KB.

LOW_MEMORY_MGR configuration parameter

Use the LOW_MEMORY_MGR configuration parameter to enable automatic low memory management, which you can use to change the default behavior of a primary or standard server when it reaches its memory limit.

onconfig.std value

LOW_MEMORY_MGR 0

values

`1` = Enables automatic low memory management when the database server starts.

`0` = Disables automatic low memory management.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

If you configure a primary or standard server to use a percentage of the SHMTOTAL configuration parameter value for automatic low memory management start and stop thresholds, the SHMTOTAL configuration parameter must be set to a positive integer value.

Attention: Changing the value of the SHMTOTAL configuration parameter can cause the configuration of automatic low memory management to become invalid, forcing the database server to use default settings.

To enable automatic low memory management, specify:

```
LOW_MEMORY_MGR 1
```

LOW_MEMORY_RESERVE configuration parameter

Use the LOW_MEMORY_RESERVE configuration parameter to reserve a specific amount of memory for use when critical activities are needed and the server has limited free memory.

If you enable the new LOW_MEMORY_RESERVE configuration parameter by setting it to a specified value in kilobytes, critical activities, such as rollback activities, can complete even when you receive out-of-memory errors.

onconfig.std value

LOW_MEMORY_RESERVE 0

values

0 or 128 - 2147483648, although the maximum value cannot be higher than 20 percent of the value of the SHMVIRTSIZE configuration parameter

units

kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

No matter how the LOW_MEMORY_RESERVE configuration parameter is set, the maximum size of reserved memory is 20 percent of the value of the SHMVIRTSIZE configuration parameter.

For example, to reserve 512 kilobytes of memory, specify:

```
LOW_MEMORY_RESERVE 512
```

You can use the onstat -g seg command to view low-memory reserve information. The output includes lines that show the size of reserved memory, the number of times that the server has used the reserved memory, and the maximum memory needed.

LTXEHWM configuration parameter

Use the LTXEHWM configuration parameter to specify the *long-transaction, exclusive-access, high-watermark*. When the logical-log space reaches the LTXEHWM threshold, the long transaction currently being rolled back is given *exclusive* access to the logical log.

onconfig.std value

LTXEHWM 80

if not present

`90` (if DYNAMIC_LOGS is set to `1` or `2`) `60` (if DYNAMIC_LOGS is set to `0`)

range of values

LTXHWM through 100

units

Percent

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

A *transaction* is *long* if it is not committed or rolled back when it reaches the long-transaction high-watermark.

If your system runs out of log space before the rollback completes, lower the LTXEHWM value.

If you do not want too many logical logs to be added, LTXEHWM should be set to a smaller value (around 60). If dynamic logging is turned off (DYNAMIC_LOGS = 0), LTXEHWM should be set lower (around 50) to avoid running out of logical space.

Tip: To allow users to continue to access the logical logs, even during a long transaction rollback, set LTXEHWM to 100. Set DYNAMIC_LOGS to 1 or 2 so that the database server can add a sufficient number of log files to prevent long transactions from hanging and to allow long transactions to roll back.

LTXHWM configuration parameter

Use the LTXHWM configuration parameter to specify the long-transaction high-watermark. The *long-transaction high-watermark* is the percentage of available log space that, when filled, triggers the database server to check for a long transaction.

onconfig.std value

LTXHWM 70

if not present

`80` (if DYNAMIC_LOGS is set to `1` or `2`) `50` (if DYNAMIC_LOGS is set to `0`)

values

1 - 100

units

Percent

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

When the logical-log space reaches the LTXHWM threshold, the database server starts rolling back the transaction. If you decrease the LTXHWM value, increase the size or number of log files to make rollbacks less likely.

If DYNAMIC_LOGS is set to `1` or `2`, the database server can add a sufficient number of log files to complete the transactions or to prevent rollbacks from hanging when you have long transactions.

If you do not want too many logical logs to be added, LTXHWM should be set to a smaller value (around 60). If dynamic logging is turned off (DYNAMIC_LOGS = 0), LTXHWM should be set lower (around 50) to avoid running out of logical space.

Warning: If you set both LTXHWM and LTXEHWM to 100, long transactions are never aborted. Although you can use this configuration to your advantage, you should set LTXHWM to below 100 for normal database server operations.

If you set LTXHWM to `100`, the database server issues a warning message:

```
LTXHWM is set to 100%. This long transaction high water mark
will never be reached. Transactions will not be aborted automatically
by the server, regardless of their length.
```

If the transaction hangs, follow the instructions for recovering from a long transaction hang, in the chapter on managing logical-log files in the *HCL OneDB™ Administrator's Guide*.

MAX_FILL_DATA_PAGES configuration parameter

Use the MAX_FILL_DATA_PAGES configuration parameter to control inserting more rows to pages that have variable-length rows.

onconfig.std value

MAX_FILL_DATA_PAGES 0

values

`0` or `1`

units

Integer

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Set the MAX_FILL_DATA_PAGES value to `1` to allow more rows to be inserted per page in tables that have variable-length rows. This setting can reduce disk space, make more efficient use of the buffer pool, and reduce table scan times.

If MAX_FILL_DATA_PAGES is enabled, the server will add a new row to a recently modified page with existing rows if adding the row leaves at least 10 percent of the page free for future expansion of all the rows in the page. If MAX_FILL_DATA_PAGES is not set, the server will add the row only if there is sufficient room on the page to allow the new row to grow to its maximum length.

A possible disadvantage of enabling MAX_FILL_DATA_PAGES and allowing more variable-length rows per page is that the server might store rows in a different physical order. Also, as the page fills, updates made to the variable-length columns in a row could cause the row to expand so it no longer completely fits on the page. This causes the server to split the row onto two pages, increasing the access time for the row.

To take advantage of this setting, existing tables with variable-length rows must be reloaded or existing pages must be modified, followed by further inserts.

MAX_INCOMPLETE_CONNECTIONS configuration parameter

Use the MAX_INCOMPLETE_CONNECTIONS configuration parameter to specify the maximum number of incomplete connections in a session.

onconfig.std value

MAX_INCOMPLETE_CONNECTIONS 1024

units

Number of incomplete connections

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

After the number specified in the MAX_INCOMPLETE_CONNECTIONS configuration parameter is reached, an error message is written in the online message log stating that the server might be under a Denial of Service attack. See also information about the LISTEN_TIMEOUT configuration parameter, which specifies the number of seconds the server waits for a connection. .

Depending on the machine capability of holding the threads (in number), you can configure MAX_INCOMPLETE_CONNECTIONS to a higher value. Depending on the network traffic, you can also set the LISTEN_TIMEOUT configuration parameter, which specifies the number of seconds the server waits for a connection, to a lower value to reduce the chance that an attack can reach the maximum limit.

MAX_PDQPRIORITY configuration parameter

Use the MAX_PDQPRIORITY configuration parameter to limit the PDQ resources that the database server can allocate to any one DSS query.

onconfig.std value

MAX_PDQPRIORITY 100

values

0 = Turns off PDQ. DSS queries use no parallelism.

1 = Fetches data from fragmented tables in parallel (parallel scans) but uses no other form of parallelism.

2 - 100 = Sets the percentage of the user-requested PDQ resources actually allocated to the query. 100 uses all available resources for processing queries in parallel.

takes effect

On all user sessions after you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

MAX_PDQPRIORITY is a factor that is used to scale the value of PDQ priority set by users. For example, suppose that the database administrator sets MAX_PDQPRIORITY to `80`. If a user sets the **PDQPRIORITY** environment variable to `50` and then issues a query, the database server silently processes the query with a PDQ priority of `40`.

You can use the onmode utility to change the value of MAX_PDQPRIORITY while the database server is online.

In HCL OneDB™, PDQ resources include memory, CPU, disk I/O, and scan threads. MAX_PDQPRIORITY lets the database administrator run decision support concurrently with OLTP, without a deterioration of OLTP performance. However, if MAX_PDQPRIORITY is too low, the performance of decision-support queries can degrade.

MIRROR configuration parameter

Use the MIRROR configuration parameter to enable or disable mirroring for the database server.

onconfig.std value

MIRROR 0

values

`0` = Disable mirroring

`1` = Enable mirroring

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

It is recommended that you mirror the root dbspaces and the critical data as part of initialization. Otherwise, leave mirroring disabled. If you later decide to add mirroring, you can edit your configuration file to change the parameter value.

You do not have to set the MIRROR configuration parameter to the same value on both database servers in the high-availability data-replication pair. You can enable or disable mirroring on either the primary or the secondary database server independently. Do not set the MIRROR configuration parameter to 1 unless you are using mirroring.

MIRROROFFSET configuration parameter

In HCL OneDB™, MIRROROFFSET specifies the offset into the disk partition or into the device to reach the chunk that serves as the mirror for the initial chunk of the root dbspace.

onconfig.std value

MIRROROFFSET 0

values

Any value greater than or equal to 0

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

MIRRORPATH configuration parameter

Use the MIRRORPATH configuration parameter to specify the full path name of the mirrored chunk for the initial chunk of the root dbspace.

onconfig.std value

On UNIX™: `$ONEDB_HOME/tmp/demo_on.root_mirror`

On Windows™: None

values

65 or fewer characters

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The MIRRORPATH should be a link to the chunk path name of the actual mirrored chunk for the same reasons that ROOTPATH is specified as a link. Similarly, select a short path name for the mirrored chunk.

You must set the permissions of the file that MIRRORPATH specifies to `660`. The owner and group must both be **informix**.

If you use raw disk space for your mirror chunk on a UNIX™ platform, it is recommended that you define MIRRORPATH as a path name that is a link to the initial chunk of the mirror dbspace, instead of entering the actual device name for the initial chunk.

To start mirroring data on a database server that is not running with the mirroring function enabled:

1. Take the database server offline.
2. Change the MIRROR configuration parameter to `1` and leave the MIRRORPATH configuration parameter blank.
3. Bring the database server online.
4. Allocate disk space for the mirror chunks. You can allocate this disk space at any time, however, the disk space must be available when you specify mirror chunks in the next step. The mirror chunks must be on a different disk than the corresponding primary chunks.
5. Specify the onspaces -m option to start mirroring for a dbspace, blobspace, or sbspace. You must begin with the root dbspace. After the root dbspace command is successfully run, the MIRRORPATH value is set automatically by the server.

MSG_DATE configuration parameter

Use the MSG_DATE configuration parameter to enable the insertion of a date in MM/DD/YY format at the beginning of each message printed to the online log.

onconfig.std value

Not in the `onconfig.std` file.

values

`0` = OFF (the default)

`1` = ON

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

In the following example MSG_DATE is set to `1` (ON).

```
04/10/11 10:26:06  Value of MSG_DATE has been changed to 1.
04/10/11 10:27:35  Value of MSG_DATE has been changed to 1.
```

MSGPATH configuration parameter

Use the MSGPATH configuration parameter to specify the full path name of the message-log file. The database server writes status messages and diagnostic messages to this file during operation.

onconfig.std value

On UNIX™: `$ONEDB_HOME/tmp/online.log`

On Windows™: `%ONEDB_HOME%\online.log`

On Windows™, if you create a server instance during installation: `%ONEDB_HOME%\`*`server_name`*`.log`. The *server_name* is the name of server in the program group and the value of the **ONEDB_SERVER** environment variable.

values

The path name of the `online.log` file.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If the file that MSGPATH specifies does not exist, the database server creates the file in the specified directory. If the directory that MSGPATH specifies does not exist, the database server sends the messages to the system console.

If the file that MSGPATH specifies does exist, the database server opens it and appends messages to it as they occur.

MULTIPROCESSOR configuration parameter

Use the MULTIPROCESSOR configuration parameter to specify whether the database server performs locking in a manner that is suitable for a single-processor computer or a multiprocessor computer.

If MULTIPROCESSOR is set to `0`, the parameters that set processor affinity are ignored.

onconfig.std value

MULTIPROCESSOR 0

values

`0` = No multiprocessor

`1` = Multiprocessor available

takes effect

After you edit your `onconfig` file and restart the database server.

NET_IO_TIMEOUT_ALARM configuration parameter

Use the NET_IO_TIMEOUT_ALARM configuration parameter to control whether to be notified if network write operations have been blocked for 30 minutes or more.

Blocked network write operations usually indicate an operating system problem. Use the NET_IO_TIMEOUT_ALARM configuration parameter to enable event alarm 82 for specific types of network traffic.

onconfig.std value

Not in `onconfig.std`

values

One of the following values or a sum of one or more of the following values:

- 0 = Disabled
- 1 = Enabled for Enterprise Replication operations
- 2 = Enabled for distributed queries
- 4 = Enabled for HDR operations
- 8 = Enabled for SMX operations
- 16 = Enabled for other component operations

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

NETTYPE configuration parameter

Use the NETTYPE parameter to tune the network protocols that are defined in the `sqlhosts` information.

onconfig.std values

UNIX™: ipcshm,1,50,CPU

Windows™: Not set.

default value

connection_type,1,50,*vp_class*

The default connection type depends on the operating system:

- UNIX™: The value of the **protocol** field from the `sqlhosts` file.
- Windows™: `onsoctcp`

The default type of virtual processor class depends on the **dbservername** entry in the `sqlhosts` file:

- CPU, if the **dbservername** `sqlhosts` entry is defined by the DBSERVERNAME configuration parameter.
- NET, if the **dbservername** `sqlhosts` entry is defined by the DBSERVERALIASES configuration parameter.

separators

Separate fields with commas. Do not include blank spaces. If you can omit values for fields, but you must include a comma for each field. However, you can omit trailing commas.

values

See the Usage section.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The NETTYPE parameter provides tuning options for the protocol and interface combinations that are associated with **dbservername** entries in the `sqlhosts` information. Each **dbservername** entry in the `sqlhosts` information is defined on either the DBSERVERNAME configuration parameter or the DBSERVERALIASES configuration parameter in the `onconfig` file.

NETTYPE *connection_type* , [{ 1 | *poll_threads* }] , [{ 50 | *conn_per_thread* }] , [{ **CPU** | **NET** }]

Table 70. Options for the NETTYPE configuration parameter value

Field	Values
connection_type	A valid protocol and interface combination, with or without the database server prefix of **on**, **ol**, or **dr**.
poll_threads	The number of poll threads that are assigned to the connection type. Default is 1. The range of values depends on the operating system and the virtual processor class: • UNIX™: If the virtual processor class type is NET, an integer greater than or equal to `1`. Each poll thread requires a separate virtual processor, so you indirectly specify the number of networking virtual processors when you specify the number of poll threads for an interface/protocol combination and specify that they are to be run by a network VP. • UNIX™: If the virtual processor class is CPU, an integer from `1` through the number of CPU VPs. • Windows™: An integer greater than or equal to `1`. If your database server has many connections, you might be able to improve performance by increasing the number of poll threads. In general, each poll thread can handle approximately 200 - 250 connections. Windows: If you specify the soctcp protocol, only one poll thread is created, and instead, a socket I/O thread (soctcpio) is created in its own SOC VP for each poll thread that is specified by the NETTYPE parameter. Socket IO threads handle receive operations for all connections using I/O completion ports to receive completion notifications. These threads perform the bulk of the work of servicing network connections on Windows™ platforms.
conn_per_thread	An integer from 1 - 32767 that sets the maximum number of connections for each poll thread. Default is 50. For shared memory connections, the value of *conn_per_thread* is the maximum number of connections per thread. In general, specify double the number of expected connections. For network connections, the value of *conn_per_thread* can be exceeded. Poll threads dynamically reallocate resources to support more connections, as needed. Avoid setting the value for the number of concurrent connections much higher than you expect. Otherwise, you might waste system resources. If only a few connections are using a protocol concurrently, you might save memory by explicitly setting the estimated number of connections.

Table 70. Options for the NETTYPE configuration parameter value

(continued)

Field	Values
CPU	Specifies a CPU virtual processor. Configure shared memory connections to run in every CPU virtual processor.
NET	Specifies to use the appropriate network virtual processor: SOC, STR, SHM, or TLI. Configure network connection to run in network virtual processors.

You can specify a NETTYPE parameter for each protocol that you want the database server to use.

The following example illustrates NETTYPE parameters for two types of connections to the database server: a shared memory connection for local clients, and a network connection that uses sockets:

```
NETTYPE ipcshm,3,,CPU
NETTYPE soctcp,8,300,NET
```

The NETTYPE parameter for the shared-memory connection (ipcshm) specifies three poll threads to run in CPU virtual processors. The number of connections is not specified, so it is set to `50`. For ipcshm, the number of poll threads correspond to the number of memory segments.

The NETTYPE parameter for the sockets connection (soctcp) specifies that 300 simultaneous connections are expected per thread for this protocol, and that 8 poll threads run in a network virtual processor.

UNIX™: There can be a dependency between the NETTYPE and NUMFDSERVERS configuration parameter settings. When you have multiple CPU virtual processors and poll threads, and thread-status output from the onstat -g ath command indicates network shared file (NSF) locking, you can increase the NUMFDSERVERS value for poll threads to reduce NSF lock contention.

NS_CACHE configuration parameter

Use the NS_CACHE configuration parameter to define the maximum retention time for entries in the HCL OneDB™ name service caches: the host name/IP address cache, the service cache, the user cache, and the group cache.

onconfig.std value

NS_CACHE host=900,service=900,user=900,group=900,sqlhosts=900

values

Each of the fields takes an integer value equal to or greater than 0.

host = Sets the number of seconds to cache information in the host name or IP address cache.

service = Sets the number of seconds to cache information in the service cache.

user = Sets the number of seconds to cache information in the user cache.

group = Sets the number of seconds to cache information in the group cache.

sqlhosts = Sets the number of seconds to cache information in the sqlhosts cache.

0 = Caching is disabled. The server always gets information from the operating system. You can set an individual cache to 0 or set all name service caches to 0: `NS_CACHE 0`.

units

Seconds

separators

Separate values with a comma. Do not include blank spaces.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

For looking up and resolving host names (or IP addresses), service names, users (and passwords) or groups, the database server queries the operating system using appropriate system calls. Similarly, the information from the sqlhosts file is read every time it is needed. You can avoid many of these lookups and file reads by using the HCL OneDB™ name service caching mechanism, which can keep and reuse each retrieved piece of information for a configurable amount of time. You should set the NS_CACHE configuration parameter if your operating system does not provide its own caching.

For the sqlhosts cache, the file cache of the operating system can be an advantage, but the database server should benefit from the sqlhosts cache more as the open()/read()/close() can be a load for the operating system in a highly concurrent environment.

The server can get information from the cache faster than it does when querying the operating system. However, if you disable one or more of these caches by setting the retention time to 0, the database server queries the operating system for the host, service, user or group information and uses direct access to the sqlhosts file.

Changes that are made to name services at the operating system level are not immediately reflected in the HCL OneDB™ name server caches: for example, the change of an IP address, a user added to or removed from a group, or a new password. However, you can use the onmode -wf or onmode -wm command to change NS_CACHE information immediately. When you change the value for a particular cache with the onmode -wf or onmode -wm command, the server immediately expires all existing entries in that cache.

NUMFDSERVERS configuration parameter

For network connections on UNIX™, use the NUMFDSERVERS configuration parameter to specify the maximum number of poll threads to handle network connections migrating between HCL OneDB™ virtual processors (VPs).

Specifying NUMFDSERVERS information is useful if HCL OneDB™ has a high rate of new connect and disconnect requests or if you find a high amount of contention between network shared file (NSF) locks. You can use the onstat -g ath command to display information about all threads. This information includes a status, such as `mutex wait nsf.lock`, which indicates that you have a significant amount of NSF lock contention.

onconfig.std value

NUMFDSERVERS 4 (Only the first 4 poll threads of each **nettype** are involved in managing the connection migrations.)

values

1 - 50

The actual number depends on the number of poll threads, which you specify in the NETTYPE configuration parameter.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The specified value of NUMFDSERVERS has no effect on shared-memory (SHM) connections.

If you use the NUMFDSERVERS configuration parameter, also review, and if necessary, change the number of poll threads in the NETTYPE configuration parameter. For example, if you have multiple CPU VPs and poll threads and this results in NSF locking, you can increase NUMFDSERVERS and poll threads to reduce NSF lock contention.

OFF_RECVRY_THREADS configuration parameter

Use the OFF_RECVRY_THREADS configuration parameter to specify the number of recovery threads that are used for logical recovery during a cold restore or fast recovery.

onconfig.std value

OFF_RECVRY_THREADS 10

values

Positive integers

units

Number of recovery threads that run in parallel

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Before you perform a cold restore, you can set the value of this parameter to approximately the number of tables that have many transactions against them in the logical log. For single-processor computers or nodes, more than 30 to 40 threads might be too many because the cost of thread management and memory offsets the increase in parallel processing.

Whenever logical recovery begins, the database server creates an LGR memory pool for the recovery threads. The size of the LGR memory pool is approximately equal to the value of OFF_RECVRY_THREADS * 100 KB. This pool is used during fast recovery and during cold restores. Do not set the OFF_RECVRY_THREADS configuration parameter to a value that results in the database server attempting to allocate more memory for the LGR memory pool than is available on your system.

In a high-availability cluster, a secondary server is almost always in fast recovery mode. On secondary servers, set the OFF_RECVRY_THREADS configuration parameter to a value that takes both roll-forward performance and memory usage into account.

ON_RECVRY_THREADS configuration parameter

The ON_RECVRY_THREADS configuration parameter is the maximum number of recovery threads that the database server uses for logical recovery when the database server is online (during a warm restore).

onconfig.std value

ON_RECVRY_THREADS 1

values

Positive integers

units

Number of recovery threads that run in parallel

takes effect

After you edit your `onconfig` file and restart the database server.

refer to

- *HCL OneDB™ Backup and Restore Guide*
- *HCL OneDB™ Performance Guide*

Usage

You can tune ON_RECVRY_THREADS to the number of tables that are likely to be recovered, because the logical-log records that are processed during recovery are assigned threads by table number. The maximum degree of parallel processing occurs when the number of recovery threads matches the number of tables being recovered.

To improve the performance of warm restores, increase the number of fast-recovery threads with the ON_RECVRY_THREADS parameter.

ONDBSPACEDOWN configuration parameter

Use the ONDBSPACEDOWN configuration parameter to define the action that the database server takes when any disabling event occurs on a primary chunk within a noncritical dbspace.

onconfig.std value

ONDBSPACEDOWN 2

values

`0` = The database server marks the dbspace as offline and continues.

`1` = The database server aborts.

`2` = The database server writes the status of the chunk to the logs and waits for user input. If you set this option, but you want the database server to mark a disabled dbspace as down and continue processing, use onmode -O to override this ONDBSPACEDOWN setting.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Database Server Behavior When ONDBSPACEDOWN Does Not Apply

The database server will not come online if a chunk within any **critical** dbspace (for example, rootdbs or logsdbs) is missing.

The value of ONDBSPACEDOWN has no effect on temporary dbspaces. For temporary dbspaces, the database server continues processing regardless of the ONDBSPACEDOWN setting. If a temporary dbspace requires fixing, you should drop and recreate it.

For a non-primary chunk within a noncritical dbspace, the behavior of the database server depends on the transaction status of the chunk when the disabling event occurs:

- **No transaction**: If no transactions are detected against that chunk, the chunk is individually marked as down. In this case, subsequent attempts to write to that chunk fail, rolling back the associated transaction. You can safely put the chunk back and then use the **onspaces -s** utility to mark the chunk as back online.
- **Transaction detected**: If there are transactions to roll forward or back, then the database server aborts with an appropriate fast recovery error. In this case, you should put the chunk back and restart the database server.

ONLIDX_MAXMEM configuration parameter

Use the ONLIDX_MAXMEM configuration parameter to limit the amount of memory that is allocated to a single *preimage* pool and a single *updator* log pool.

onconfig.std value

ONLIDX_MAXMEM 5120

values

16 - 4294967295

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The preimage and updator log pools, **pimage**_*partnum* and **ulog**_*partnum*, are shared memory pools that are created when a CREATE INDEX ONLINE statement is executed. The pools are freed when the execution of the statement is completed.

If you specify a value for this parameter and then create a table, add rows to the table, and start to execute a CREATE INDEX ONLINE statement on a column, you can also perform other operations on the column, such as running UPDATE STATISTICS HIGH, without having memory problems.

OPCACHEMAX configuration parameter (UNIX™)

onconfig.std value

OPCACHEMAX 0

if not present

`128`

values

0 - `(4 * 1024 * 1024)`

units

Kilobytes

takes effect

When the Optical Subsystem needs more memory

Usage

OPCACHEMAX specifies the size of the memory cache for the Optical Subsystem. The database server stores pieces of TEXT or BYTE data in the memory cache before it delivers them to the subsystem. Use this parameter only if you use the Optical Subsystem.

The **IONEDB_ OPCACHE** environment variable lets the client restrict the size of the optical cache that it uses.

OPTCOMPIND configuration parameter

Use the OPTCOMPIND to specify information that helps the optimizer choose an appropriate query plan for your application.

Tip: You can think of the name of the variable as arising from OPTimizer COMPare (the cost of using) INDexes (with other methods).

onconfig.std value

OPTCOMPIND 2

values

`0` = When appropriate indexes exist for each ordered pair of tables, the optimizer chooses index scans (nested-loop joins), without consideration of the cost, over table scans (hash joins). This value ensures compatibility with previous versions of the database server.

`1` = The optimizer uses costs to determine an execution path if the isolation level is not Repeatable Read. Otherwise, the optimizer chooses index scans (it behaves as it does for the value `0`). This setting is recommended for optimal performance.

`2` = The optimizer uses cost to determine an execution path for any isolation level. Index scans are not given preference over table scans; the optimizer bases its decision purely on cost. This value is the default if the variable is not set.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Because of the nature of *hash joins*, an application with isolation mode set to Repeatable Read might *temporarily* lock all records in tables that are involved in the join (even those records that fail to qualify the join) for each ordered set of tables. This situation leads to higher contention among connections. Conversely, nested-loop joins lock fewer records but provide inferior performance when the database server retrieves a large number of rows. Thus, both join methods offer advantages and disadvantages. A client application can also influence the optimizer in its choice of a join method.

OPT_GOAL configuration parameter

Use the OPT_GOAL configuration parameter to specify an optimization goal for queries.

onconfig.std value

OPT_GOAL -1

values

0 or -1

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

A value of `0` sets the optimization goal to FIRST_ROWS. A value of `-1` sets the optimization goal to ALL_ROWS, which is the default.

When you set the optimization goal to optimize for FIRST ROWS, you specify that you want the database server to optimize queries for perceived response time. In other words, users of interactive applications perceive response time as the time that it takes to display data on the screen. Setting the optimization goal to FIRST ROWS configures the database server to return the first rows of data that satisfy the query.

When you set the optimization goal to optimize for ALL ROWS, you specify that you want the database server to optimize for the total execution time of the query. Making ALL ROWS the optimization goal instructs the database server to process the total query as quickly as possible, regardless of how long it takes to return the first rows to the application.

You can specify the optimization goal in one of four ways:

- By query (SELECT statement)

  Use the ALL_ROWS and FIRST_ROWS directives.
- By session

  Use the SET OPTIMIZATION statement.
- By environment

  Set the **OPT_GOAL** environment variable.
- By database server

  Set the OPT_GOAL configuration parameter.

The list above lists the mechanisms for setting this goal in descending order of precedence. To determine the optimization goal, the database server examines the settings in the order above. The first setting encountered determines the optimization goal. For example, if a query includes the ALL_ROWS directive but the OPT_GOAL configuration parameter is set to FIRST_ROWS, the database server optimizes for ALL_ROWS, as the query specifies.

PC_HASHSIZE configuration parameter

Use PC_HASHSIZE to specify the number of hash buckets in the caches that the database server uses. PC_HASHSIZE applies to UDR cache only.

onconfig.std value

PC_HASHSIZE 31

values

Any positive integer, a prime number is recommended.

takes effect

After you edit your `onconfig` file and restart the database server.

PC_POOLSIZE configuration parameter

Use the PC_POOLSIZE configuration parameter to specify the maximum number of user-defined routines that are stored in the UDR cache.

onconfig.std value

PC_POOLSIZE 127

values

A positive value 127 or greater that represents the maximum number of entries in the cache. A positive value 127 or greater that represents half of the initial maximum number of entries in the cache. The maximum value is dependent upon the shared memory configuration and available shared memory for the server instance.

takes effect

After you edit your `onconfig` file and restart the database server.

When you increase the value in memory by running the onmode -wm command.

When you reset the value in memory by running the onmode -wm command.

The initial number of entries in the cache is twice the value of the PC_POOLSIZE configuration parameter. For example, if the PC_POOLSIZE configuration parameter is set to 127, 254 entries are allowed in the cache. If all entries in the cache are full, the cache size automatically grows by 10%. To reduce the size of the cache, decrease the value of the PC_POOLSIZE configuration parameter in the `onconfig` file and restart the server.

PHYSBUFF configuration parameter

Use the PHYSBUFF configuration parameter to specify the size in kilobytes of the two physical-log buffers in shared memory.

onconfig.std value

PHYSBUFF 128

units

Kilobytes

values

An integer in the range of 4 - (32767 * *pagesize* / 1024), where *pagesize* is the default system page size. The value must be evenly divisible by the default system page size. If the value is not evenly divisible by the page size, the database server rounds down the size to the nearest value that is evenly divisible by the page size.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Double buffering permits user threads to write to the active physical-log buffer while the other buffer is being flushed to the physical log on disk. A write to the physical-log buffer is exactly one page in length. The value of the PHYSBUFF parameter determines how frequently the database server needs to flush the physical-log buffer to the physical-log file.

If the RTO_SERVER_RESTART configuration parameter is enabled, use the 512 kilobyte default value for PHYSBUFF. If the value of the PHYSBUFF configuration parameter is less than 512 kilobytes when the RTO_SERVER_RESTART configuration parameter is enabled, a warning message displays when you restart the server.

The user-data portion of a smart large object does not pass through the physical-log buffers.

PHYSFILE configuration parameter

Use the PHYSFILE configuration parameter to specify the size of the physical log file when you first initialize the disk space and bring the database server online.

onconfig.std value

PHYSFILE 50000

if not present

`200`

values

An integer 200 or greater

units

KB

takes effect

After you edit the `onconfig` file and initialize disk space by running the oninit -i command.

After you run the onparams -p -s command.

Usage

You cannot change the value of the PHYSFILE configuration parameter by editing the `onconfig` file after you start the server for the first time.

The database server updates the value of the PHYSFILE configuration parameter in the `onconfig` file under the following circumstances:

- You change the size of the physical log file by running the onparams -p -s command.
- The plogspace is automatically expanded. If the physical log is stored in a plogspace, the database server expands the size of the physical log as needed to improve performance.

The database server updates the value of the PHYSFILE configuration parameter in the `onconfig` file when you change the size of the physical log file by running the onparams -p -s command.

When the RTO_SERVER_RESTART or SEC_NONBLOCKING_CKPT configuration parameter is enabled, ensure that the size of the physical log is equal to at least 110% of the buffer pool size.

A warning message prints to the message log when:

- The value for the PHYSFILE configuration parameter is changed to less than 110% of all of the buffer pools
- The server is restarted
- A new buffer pool is added

PLOG_OVERFLOW_PATH configuration parameter

The PLOG_OVERFLOW_PATH parameter specifies the location of the file that is used during fast recovery if the physical log file overflows.

The file is `plog_extend.`*`servernum`* and by default located in `$ONEDB_HOME/tmp`. Use the full path name to specify a different location for the file with the PLOG_OVERFLOW_PATH parameter.

onconfig.std values

On UNIX™: `$ONEDB_HOME/tmp`

On Windows™: None

takes effect

When the database server is brought up (shared memory is initialized)

PLCY_HASHSIZE configuration parameter

The PLCY_HASHSIZE configuration parameter specifies the number of hash buckets in the security policy information cache.

onconfig.std value

PLCY_HASHSIZE 31

values

Any positive integer

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

PLCY_POOLSIZE configuration parameter

Use the PLCY_POOLSIZE configuration parameter to specify the maximum number of entries in each hash bucket of the security policy information cache.

onconfig.std value

PLCY_POOLSIZE 127

values

A positive value 127 or greater that represents the maximum number of entries in the cache. A positive value 127 or greater that represents half of the initial maximum number of entries in the cache. The maximum value is dependent upon the shared memory configuration and available shared memory for the server instance.

takes effect

After you edit your `onconfig` file and restart the database server.

When you increase the value in memory by running the onmode -wm command.

When you reset the value in memory by running the onmode -wm command.

The initial number of entries in the cache is twice the value of the PLCY_POOLSIZE configuration parameter. For example, if the PLCY_POOLSIZE configuration parameter is set to 127, 254 entries are allowed in the cache. If all entries in a cache are full, the cache size automatically grows by 10%. To reduce the size of the cache, decrease the value of the PLCY_POOLSIZE configuration parameter in the `onconfig` file and restart the server.

PN_STAGEBLOB_THRESHOLD configuration parameter

Use the PN_STAGEBLOB_THRESHOLD configuration parameter to reserve space for BYTE and TEXT data in round-robin fragments.

onconfig.std value

Not set.

if not present

0

values

0 - 1000000

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Set this configuration parameter to the typical or average size of the BYTE or TEXT data that is stored in the table.

Restriction: The PN_STAGEBLOB_THRESHOLD configuration parameter has no effect if the number of extents has reached the maximum extents allowed or if the dbspace is full.

When a table reaches the maximum number of pages for a fragment, more pages can be added to the table by adding a new fragment. However, if a table contains BYTE or TEXT columns and that table is fragmented by the round-robin distribution scheme, adding a new fragment does not automatically enable new rows to be inserted into the new fragment.

For example, if one of the fragments in the table reaches the maximum number of pages, adding a new fragment does not extend the table to store more rows. Because BYTE and TEXT data tend to be large in size, the data is *staged* in one of the fragments before being distributed evenly in all of the fragments. The staging fragment must have sufficient space to store the BYTE or TEXT data. Use the PN_STAGEBLOB_THRESHOLD configuration parameter so that the database server can stage the BYTE or TEXT data temporarily in a staging fragment until the INSERT operation is completed and the data is permanently stored in the table.

During a UPDATE operation if the fragment does not have the space that is specified in PN_STAGEBLOB_THRESHOLD configuration parameter the table row that is impacted by the updated is moved into another fragment.

PRELOAD_DLL_FILE configuration parameter

The PRELOAD_DLL_FILE configuration parameter specifies the path name for a shared library file that is preloaded when the database server is started.

onconfig.std value

Not set. No shared library files are preloaded.

value

pathname = Full path name for the shared library file. Can include $ONEDB_HOME.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Use this parameter to preload the shared library files for DataBlade® modules, built-in extensions, or user-defined routines that are created in the C programming language (C UDRs). Otherwise, the shared libraries are loaded when they are first used after the server starts, which affects performance. Add a separate entry of this parameter for each library file that you want to preload. A preloaded shared library remains active until the server is stopped.

Restriction: You cannot use the onmode -wm or onmode -wf commands to set the PRELOAD_DLL_FILE configuration parameter.

Example

Examples

The following examples preload the built-in basic text search, spatial, and time series extensions:

```
PRELOAD_DLL_FILE $ONEDB_HOME/extend/bts.version/bts.bld

PRELOAD_DLL_FILE $ONEDB_HOME/extend/TimeSeries.version/TimeSeries.bld
```

The *version* is the specific version number for the extension. To find the correct version number, run the appropriate function to return the release number for the extension or check the directory name in your installation directory.

Important: The version numbers of built-in extensions can change in any fix pack or release. After you upgrade, you must update the value of the PRELOAD_DLL_FILE configuration parameter if the version number of an extension changed.

QSTATS configuration parameter

The QSTATS configuration parameter specifies the ability of onstat -g qst to print queue statistics.

onconfig.std value

QSTATS 0

values

`0` = Disable queue statistics

`1` = Enable queue statistics

takes effect

After you edit your `onconfig` file and restart the database server.

REMOTE_SERVER_CFG configuration parameter

Use the REMOTE_SERVER_CFG configuration parameter to specify the file that lists trusted remote hosts.

onconfig.std value

Not set. The system `hosts.equiv` file is used.

values

File name. The path is assumed to be `$ONEDB_HOME/etc`. Consider using the following naming convention:

```
authfile.server_name
```

The file that is specified by the REMOTE_SERVER_CFG configuration parameter must be in `$ONEDB_HOME/etc`.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

For applications that connect to the database server as the **root** user, the REMOTE_SERVER_CFG configuration parameter is not applicable.

The format of the file that is specified by the REMOTE_SERVER_CFG configuration parameter is the same as the format of the system `hosts.equiv` file.

If the REMOTE_SERVER_CFG configuration parameter is not set, and you run the SQL administration API task() or admin() function with the cdr add trustedhost argument, the database server performs the following actions:

1. The REMOTE_SERVER_CFG configuration parameter is set to `authfile.`*DBSERVER*.
2. The `authfile.`*DBSERVER* file is created in `$ONEDB_HOME/etc`.
3. The specified trusted-host information is added to `$ONEDB_HOME/etc/authfile.`*DBSERVER*.
4. If the database server is part of a high-availability cluster, the trusted-host information is propagated to the trusted-host files of the other cluster servers.

Note: If the `sqlhosts` file of the database server uses the `s=6` option, you must also set the S6_USE_REMOTE_SERVER_CFG configuration parameter to `1` to use the file specified REMOTE_SERVER_CFG configuration parameter. Otherwise, the database server uses the system `hosts.equiv` file instead of the file specified REMOTE_SERVER_CFG configuration parameter.

REMOTE_USERS_CFG configuration parameter

Use the REMOTE_USERS_CFG configuration parameter to specify the file that lists the names of trusted users that exist on remote hosts.

onconfig.std value

Not set.

values

File name. The path is assumed to be `$INFORMIX/etc`.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The file specified by the REMOTE_USERS_CFG configuration parameter must be located in `$ONEDB_HOME/etc`. If the configuration parameter is set then the file specified is used instead of the `~/.rhosts` file. If the specified file does not exist in `$ONEDB_HOME/etc`, then authentication will fail.

The format of the file specified by the REMOTE_USERS_CFG configuration parameter is the same as the format of the `~/.rhosts` file.

Consider using the following naming convention for the file specified by the REMOTE_USERS_CFG configuration parameter:

```
users.server_name
```

RESIDENT configuration parameter

Use the RESIDENT configuration parameter to specify whether resident and virtual segments of shared memory remain resident in operating-system physical memory.

onconfig.std value

RESIDENT 0

values

-1 - 99

`0` = off

`1` = lock the resident segment only

`-1` = lock all resident and virtual segments

n = lock the resident segment and the next *n* -1 virtual segments. For example, if you specify `99` as the value, the resident segment is locked and the next 98 virtual segments are locked.

Certain platforms have different values. For information, see your machine notes.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Some systems allow you to specify that the resident portion of shared memory must stay (be resident) in memory at all times. If your operating system supports forced residency, you can specify that resident and virtual segments of shared memory not be swapped to disk.

Warning: Before you decide to enforce residency, verify that the amount of physical memory available is sufficient to execute all required operating-system and application processes. If insufficient memory is available, a system hang could result that requires a reboot.

On AIX®, Solaris, or Linux™ systems that support large pages of memory, the DBSA can use operating system commands to configure a pool of large pages.

HCL OneDB™ can store non-message virtual memory segments on these large pages if you take the following steps:

- Enable large page sizes by setting the **IFX_LARGE_PAGES** environment variable.
- For virtual memory segments that you intend to store on large pages, set the RESIDENT parameter to lock those segments in physical memory, so that they cannot be swapped to disk

Storing virtual memory segments on large pages can offer significant performance benefits in large memory configurations.

RESTARTABLE_RESTORE configuration parameter

Use the RESTARTABLE_RESTORE configuration parameter to control whether the database server performs restartable restores.

onconfig.std value

RESTARTABLE_RESTORE ON

values

`ON` = Restartable restore is enabled

`OFF` = Restartable restore is disabled

takes effect

After you edit your `onconfig` file and restart the database server.

If you set RESTARTABLE_RESTORE to `ON`, you enable the database server to restart a failed physical or cold logical restore at the point at which the failure occurred. To perform a restartable restore with ON-Bar, use the onbar -RESTART command.

Increase the size of your physical log if you plan to use restartable restore. Although a restartable restore slows down the logical restore if many logs need to be restored, you save a lot of time from not having to repeat the entire restore.

Important: If the database server fails during a warm logical restore, you must repeat the entire restore. If the database server is still running, use onbar -r -l to complete the restore.

If you do a cold restore on systems that are not identical, you can assign new pathnames to chunks, and you can rename devices for critical chunks during the restore. You must perform a level-0 archive after the rename and restore operation completes.

The database server uses physical recovery and logical recovery to restore data as follows:

- **Physical recovery.** The database server writes data pages from the backup media to disk. This action leaves the storage spaces consistent to the point at which it was originally backed up. However, the backup times for each storage space are usually different. A restartable restore is restartable to the level of a storage space. If only some chunks of a storage space are restored when the restore fails, the entire storage space needs to be recovered again when you restart the restore.
- **Logical recovery.** The database server replays logical-log records on media to bring all the storage spaces up to date. At the end of logical recovery, all storage spaces are consistent to the same point.

RESTORE_POINT_DIR configuration parameter

Use the RESTORE_POINT_DIR configuration parameter to change the path name of the directory where restore point files will be placed during a failed upgrade to a new version of the server. HCL OneDB™ will store restore point files in a subdirectory of the specified directory, with the server number as the subdirectory name, only if the CONVERSION_GUARD configuration parameter is enabled.

onconfig.std value

`$ONEDB_HOME/tmp`

value

Complete path name for a directory

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

You can change the directory, for example, if you think that the `$ONEDB_HOME/tmp` directory does not have enough space for restore point data. If you want to change the directory, you must change it before you initiate an upgrade to a new version of the server. You cannot change the directory during an upgrade.

The directory specified in the RESTORE_POINT_DIR configuration parameter must be empty when an upgrade begins. If the directory contains any restore point files from a previous upgrade, you must remove the files before a new upgrade begins a new restore point.

Important:

The empty directory is a prerequisite before doing the upgrade, not when recovering from a failed upgrade. After a failed upgrade, do not empty the RESTORE_POINT_DIR directory before you attempt to run the onrestorept utility.

ROOTNAME configuration parameter

ROOTNAME specifies a name for the root dbspace for this database server configuration.

The name must be unique among all dbspaces that the database server manages. It is recommended that you select a name that is easily recognizable as the root dbspace.

onconfig.std value

ROOTNAME rootdbs

values

Up to 128 bytes. ROOTNAME must begin with a letter or underscore and must contain only letters, numbers, underscores, or **$** characters.

units

A dbspace

takes effect

When disk is initialized (destroys all data)

ROOTOFFSET configuration parameter

ROOTOFFSET specifies the offset into an allocation of disk space (file, disk partition, or device) at which the initial chunk of the root dbspace begins.

UNIX Only:

On some UNIX™ platforms, it is not valid to set ROOTOFFSET to `0`. When this parameter is set incorrectly, you must reinitialize disk space and reload data to resume proper operation of the database server. Before you configure the database server, always check your machine notes file for information about proper settings.

onconfig.std value

ROOTOFFSET 0

values

Any value greater than or equal to 0

units

Kilobytes

takes effect

When disk is initialized (destroys all data)

ROOTPATH configuration parameter

Use the ROOTPATH configuration parameter to specify the full path name, including the device or file name, of the initial chunk of the root dbspace. The ROOTPATH configuration parameter is stored in the reserved pages as a chunk name.

onconfig.std value

On UNIX™: `$ONEDB_HOME/tmp/demo_on.rootdbs`
On Windows™: None

values

pathname

takes effect

When disk is initialized (destroys all data)

refer to

The following material in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*

- Allocating disk space
- Creating links for raw devices

Usage

On UNIX™, you must set the permissions of the file that you specify with the ROOTPATH configuration parameter to `660`, and the owner and group must both be **informix**. On Windows™, a member of the **Informix-Admin** group must own the file that you specify with the ROOTPATH configuration parameter.

UNIX Only:

If you use unbuffered disk space for your initial chunk on UNIX™, you should define the ROOTPATH configuration parameter as a pathname that is a link to the initial chunk of the root dbspace instead of entering the actual device name for the initial chunk.

ROOTSIZE configuration parameter

Use the ROOTSIZE configuration parameter to specify the size in kilobytes of the initial chunk of the root dbspace. The size that you select depends on your immediate plans for your database server.

The database server uses the value of the ROOTSIZE configuration parameter only during a complete disk initialization. Changing the ROOTSIZE value after the initial chunk of the root dbspace has been created will have no effect.

onconfig.std value

ROOTSIZE 200000ROOTSIZE 300000

if not present

`0`

values

50,000 through maximum capacity of the storage device

units

Kilobytes

takes effect

When disk is initialized (destroys all data)

RSS_FLOW_CONTROL configuration parameter

Specifies when flow control occurs in a high-availability cluster that contains at least one remote standalone (RS) secondary server.

onconfig.std value

RSS_FLOW_CONTROL 0

values

`0` = Flow control is activated when the difference between the current log position and the most recent acknowledged log exceeds 12 times the size of the log buffer.

`-1` = Flow control is disabled. Disabling flow control might lead to wrapping of the log files and the loss of data.

`start_value,end_value` = The *start_value* and *end_value* determine the amount of lag between the current log position and the last acknowledged log page. The *start_value* must be greater than the *end_value*. Values must include one of the following units:

- `K` (Kilobytes)
- `M` (Megabytes)
- `G` (Gigabytes)

For example, setting `RSS_FLOW_CONTROL 128M,100M` starts flow control when the lag between the logs is 128 MB, and stops flow control when the lag drops to 100 MB.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Flow control provides a way to limit log activity on the primary server so that RS secondary servers in the cluster do not fall too far behind on processing transactions. Enabling flow control ensures that logs on RS secondary servers remain current if the servers are on a busy or intermittent network. When flow control is enabled, and when the difference in log size between the current log position and the last acknowledged log page exceeds the*start_value*, then log activity on the primary server becomes restricted. Users connected to the primary server may experience slower response time when flow control is active. Flow control is started when the lag between the logs is greater than the *start_value* and stops flow control when the log lag has dropped to the*stop_value*.

You set the RSS_FLOW_CONTROL configuration parameter on the primary server only. All RS secondary servers in the cluster are affected by the RSS_FLOW_CONTROL configuration parameter. Logs are always sent to the RS secondary server in the order in which they were received.

To check if flow control is active for a RS secondary server, use the onstat -g rss verbose command, and compare the `RSS flow control` value to the `Approximate Log Page Backlog` value. If the `Approximate Log Page Backlog` is higher than the first value of `RSS flow control`, flow control is active. If the `Approximate Log Page Backlog` is lower than the second value of `RSS flow control`, flow control is disabled.

RSS_NONBLOCKING_CKPT configuration parameter

Use the **RSS_NONBLOCKING_CKPT** configuration parameter to enable non-blocking checkpoint at RS secondary server.

onconfig.std value

RSS_NONBLOCKING_CKPT 0

values

- 1 - Enable non-blocking checkpoint at RS secondary server
- 0 – (Default) Disable non-blocking checkpoints

units

Not applicable

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

RSS_NONBLOCKING_CKPT configuration parameter control the checkpoint behavior at RS secondary server.

Attention: RSS_NONBLOCKING_CKPT configuration parameter will be deprecated in future releases. User should use SEC_NONBLOCKING_CKPT instead.

Related reference

SEC_NONBLOCKING_CKPT configuration parameter on page 169

RTO_SERVER_RESTART configuration parameter

Use the RTO_SERVER_RESTART configuration parameter to specify recovery time objective (RTO) standards for the amount of time, in seconds, that HCL OneDB™ has to recover from a problem after you restart the server and bring it into online or quiescent mode.

onconfig.std value

RTO_SERVER_RESTART 0 (disabled)

range of values

`0` = disabled

60 - 1800

units

seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

S6_USE_REMOTE_SERVER_CFG configuration parameter

Use the S6_USE_REMOTE_SERVER_CFG configuration parameter to control whether the file specified by the REMOTE_SERVER_CFG configuration parameter is used to authenticate secure connections for server clusters and Enterprise Replication.

onconfig.std value

S6_USE_REMOTE_SERVER_CFG 0

default value

0

values

0 = The system `hosts.equiv` file is used to authenticate servers connecting through a secure port.

1 = The file specified by the REMOTE_SERVER_CFG configuration parameter is used to authenticate servers connecting through a secure port.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The REMOTE_SERVER_CFG configuration parameter is used to specify a file that lists the remote server hosts that are trusted by the computer housing the database server. If one or more of the listed servers are configured using the `sqlhosts` file connection-security option `s=6`, then you must set the S6_USE_REMOTE_SERVER_CFG configuration parameter to `1`.

If S6_USE_REMOTE_SERVER_CFG is unset or set to `0`, the system `hosts.equiv` file, rather than the file specified by the REMOTE_SERVER_CFG configuration parameter, is used to authenticate servers connecting through a secure port.

SB_CHECK_FOR_TEMP configuration parameter

Use the SB_CHECK_FOR_TEMP configuration parameter to prevent the copying of a temporary smart large object into a permanent table.

onconfig.std value

Not set.

if value not present

The copying of temporary smart large objects into permanent tables is permitted.

values

`0` = Permit the copying of temporary smart large objects into permanent tables. Equivalent to the configuration parameter not being set in the `onconfig` file.

`1` = Prevent the copying of temporary smart large objects into permanent tables. The database server returns the following error messages instead of copying the handle of a temporary smart large object:

- `-9810: Smart-large-object error.`
- `-12246: Smart large objects: You cannot put a temporary smart large object into a permanent table`

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

By default, you can copy temporary smart large objects into permanent tables. Smart large object data types, BLOB and CLOB, consist of two parts: the data, which is stored in an sbspace, and the handle, which is stored in a table. When you copy a temporary smart large object into a permanent table, only the BLOB or CLOB handle is copied into the permanent table. If you subsequently drop the temporary smart large object, the permanent table contains a handle that is no longer valid.

To prevent the copying of a temporary smart large object into a permanent table, set the SB_CHECK_FOR_TEMP configuration parameter to 1 in the `onconfig` file. For example, if the SB_CHECK_FOR_TEMP configuration parameter is set to 1, an INSERT INTO . . . SELECT FROM . . . statement that copies a temporary smart large object into a permanent table fails.

SBSPACENAME configuration parameter

Use the SBSPACENAME configuration parameter specifies the name of the default sbspace.

onconfig.std value

Not set.

if not present

0

values

Up to 128 bytes.

SBSPACENAME must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or **$** characters.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If your database tables include smart-large-object columns that do not explicitly specify a storage space, that data is stored in the sbspace that SBSPACENAME specifies.

The default sbspace is also used by the built-in encryption and decryption functions to store BLOB or CLOB values. If DECRYPT_BINARY or an encryption function cannot find an sbspace in which to store a BLOB or CLOB argument or returned value, the function fails with the following error message:

```
Fatal error in server row processing - SQL error -9810 ISAM error -12053
```

If you see this error message after you invoke an encryption or decryption function that has a CLOB or BLOB argument, configure a default sbspace using the SBSPACENAME configuration parameter, and then repeat the function call.

You must create the default sbspace with the onspaces -c -S utility before you can use it. The database server validates the name of the default sbspace when one of the following occurs:

- You specify the default sbspace as the storage option for a CLOB or BLOB column in the PUT clause of the CREATE TABLE or ALTER TABLE statement.
- The database server attempts to write a smart large object to the default sbspace when no sbspace was specified for the column.
- You store multirepresentational data in the default sbspace.

JAVA Language Support:

If you are using J/Foundation, you must provide a smart large object where the database server can store the Java™ archive (JAR) files. These JAR files contain your Java™ user-defined routines (UDRs). It is suggested that when you use Java™ UDRs, you create separate sbspaces for storing smart large objects.

Warning: When you use Enterprise Replication, you must set the CDR_QDATA_SBSPACE parameter and create the sbspace before you define the replication server.

Automatic creation of the default sbspace

A default sbspace is created even if the SBSPACENAME configuration parameter is not set if you create a **bts** index and do not explicitly specify an sbspace name.

The default sbspace is created in the root dbspace for the database server with a size of 10 000 KB. You must manually increase the size of the default sbspace when it fills.

SBSPACETEMP configuration parameter

Use the SBSPACETEMP configuration parameter to specify a list of default temporary sbspace for storing temporary smart large objects without metadata or user-data logging. If you store temporary smart large objects in a standard sbspace, the metadata is logged.

onconfig.std value

Not set. Temporary smart large objects are stored in the default sbspace, which is specified by the SBSPACENAME configuration parameter.

separators

Commas

values

One or more sbspace names. Separate names with a comma. The length of the list cannot exceed 128 bytes.

Each sbspace name must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or **$** characters.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

SDS_ALTERNATE configuration parameter

Use the SDS_ALTERNATE configuration parameter to define an alternate means of communication between the primary server and SD secondary servers in a high-availability cluster.

onconfig.std value

NONE (No SD secondary server alternate communication path is configured.)

values

The name of the blobspace that is to be used as the alternate communication path between the primary server and SD secondary servers.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

You set the SDS_ALTERNATE configuration parameter and create a shared blobspace to allow the primary server and all SD secondary servers in a high-availability cluster to use an alternate communication path in the event the network is unavailable between the primary server and the SD secondary servers. When an SD secondary server is about to failover and become the primary server, but TCP/IP communication is unavailable, the shared blobspace set by the SDS_ALTERNATE configuration parameter is used communicate the shut-down procedure to the original primary.

Set the SDS_ALTERNATE configuration parameter to the same value on the primary server and on all SD secondary servers.

Before setting the SDS_ALTERNATE configuration parameter, you must create the shared blobspace on the primary server. For example, to create a blobspace named sds_alt_comm enter the following command on the primary server:

```
onspaces -c -b sds_alt_comm -g <pagesize> -p <path> -o <offset> -s <size>
```

Run the following command to switch to the next logical log file so that the newly created blobspace is usable:

```
onmode -l
```

On each of the SD secondary servers in the high-availability cluster, set the SDS_ALTERNATE configuration parameter to point to the blobspace on the primary server.

```
SDS_ALTERNATE sds_alt_comm
```

SDS_ENABLE configuration parameter

Use the SDS_ENABLE configuration parameter to enable SD secondary server functionality.

onconfig.std value

Not set.

if not present

`0`

values

`0` = Disable

`1` = Enable

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

You must set SDS_ENABLE to 1 (enable) on the SD secondary server to enable SD secondary server functionality.

SDS_ENABLE is set to 1 (enabled) automatically when you run the following command:

```
onmode -d set SDS primary
```

SDS_ENABLE is set to 0 (disabled) when you run the following command:

```
onmode -d clear SDS primary
```

To prevent data corruption, you cannot use the oninit -i or oninit -iy command to initialize disk space on a server if SDS_ENABLE is set to 1 (enabled). To initialize an SD secondary server, initialize only the shared memory by using oninit with no parameters. To initialize a primary server to which one or more SD secondary servers are attached, and whose disk has never been initialized, set SDS_ENABLE to 0 and initialize the server memory and disk using oninit -i. To initialize a primary server to which SD secondary servers are attached, and whose disk is already initialized, set SDS_ENABLE to 1 and initialize shared memory only using oninit with no parameters.

SDS_FLOW_CONTROL configuration parameter

Specifies when flow control occurs in a high-availability cluster that contains at least one shared-disk (SD) secondary server.

onconfig.std value

SDS_FLOW_CONTROL 0

values

`0` = Flow control is activated when the difference between the current log position and the most recent acknowledged log exceeds 12 times the size of the log buffer.

`-1` = Flow control is disabled. Disabling flow control might lead to wrapping of the log files and the loss of data.

`start_value,end_value` = The *start_value* and *end_value* determine the amount of lag between the current log position and the last acknowledged log page. The *start_value* must be greater than the *end_value*. Values must include one of the following units:

- `K` (Kilobytes)
- `M` (Megabytes)
- `G` (Gigabytes)

For example, setting `SDS_FLOW_CONTROL 128M,100M` starts flow control when the lag between the logs is 128 MB, and stops flow control when the lag has dropped to 100 MB.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

Usage

Flow control provides a way to limit log activity on the primary server so that SD secondary servers in the cluster do not fall too far behind on processing transactions. When flow control is enabled, and when the difference in log size between the current log position and the last acknowledged log page exceeds the*start_value*, then log activity on the primary server becomes restricted. Users connected to the primary server may experience slower response time when flow control is active. Flow control is started when the lag between the logs is greater than the *start_value* and stops flow control when the log lag has dropped to the*stop_value*.

You set the SDS_FLOW_CONTROL configuration parameter on the primary server only. All SD secondary servers in the cluster are affected by the SDS_FLOW_CONTROL configuration parameter. Logs are always sent to the SD secondary server in the order in which they were received.

SDS_LOGCHECK configuration parameter

Use the SDS_LOGCHECK configuration parameter to set the number of seconds to delay the secondary server from taking over the role of the primary server. If the secondary server detects that the primary server is generating log records during the delay period, then the failover is prevented. The delay can prevent an unnecessary failover if network communication between the primary and secondary servers is temporarily unavailable.

onconfig.std value

SDS_LOGCHECK 0

SDS_LOGCHECK

On UNIX™: 10
On Windows™: 0

values

`0` = Do not detect log activity; allow immediate failover.

n = Wait up to *n* seconds. If log activity is detected from the primary server, failover is prevented; otherwise, failover is allowed.

units

Seconds

takes effect

When shared disk functionality is enabled on the primary server

Usage

Important: You must specify the same value for the primary server and for all secondary servers. If the values that you specify are not the same, the database server automatically changes the value that is different on a secondary server to the value that is set for the primary server.

For example, if the SDS_LOGCHECK configuration parameter is set to 10, and the primary server fails, the SD secondary server waits up to 10 seconds to either detect that the primary server is generating log records (in which case failover is prevented), or the SD secondary server detects that the primary is not generating log records and failover occurs.

An unnecessary failover can result in two primary servers that are both receiving input from applications and writing to the same chunks, which can cause unrepairable data corruption.

Set the SDS_LOGCHECK configuration parameter to a value greater than zero if you do not have I/O fencing configured and your system consists of a primary server and one or more SD secondary servers.

If your system has I/O fencing configured, and if an SD secondary server becomes a primary server, the I/O fencing script must prevent the failed primary server from updating any of the shared disks. If the system does not have I/O fencing configured, the SDS_LOGCHECK configuration parameter prevents the occurrence of multiple primary servers by not failing over to the SD secondary server if the original primary server is generating log records.

SDS_PAGING configuration parameter

The SDS_PAGING configuration parameter specifies the location of two files that serve as buffer paging files.

onconfig.std value

Not set

Values

File paths

Separators

A single comma

Default value

None

Takes effect

When SD secondary server is started

Usage

The SDS_PAGING configuration parameter must be set to a valid value to ensure that the SD secondary server starts. Because the paging files grow dynamically as needed, you should allocate enough disk space to store two times the size of the value specified by the PHYSFILE configuration parameter.

Example

Example

In the following example, the files `page1` and `page2` are set as the buffer paging files for the SD secondary server.

```
SDS_PAGING /usr/informix/tmp/page1,/usr/informix/tmp/page2
```

SDS_TEMPDBS configuration parameter

Use the SDS_TEMPDBS configuration parameter to specify information that the shared disk (SD) secondary server uses to dynamically create temporary dbspaces. This configuration parameter can be specified only on the SD secondary server.

onconfig.std value

Not set. Temporary dbspaces for shared disk secondary servers are not created.

values

A string containing the following values in the following order, separated by commas:

dbspace = The name of the dbspace to create. Must be unique among all existing dbspaces, blobspaces, and sbspaces, including those any temporary spaces that are inherited from a primary server. The name cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the $ character.

dbpath = The path for the dbspace, either a full path name or a relative path name. If you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server.

pagesize = An integer representing the page size of the dbspace, in kilobytes. The page size must be between 2 KB and 16 KB and must be a multiple of the default page size.

offset = An integer equal to or greater than 0 that specifies offset into the disk partition or into the device to reach the initial chunk of the dbspace. The starting offset plus the chunk size cannot exceed the maximum chunk size. The offset must be a multiple of the page size. The maximum offset is 2 or 4 terabytes, depending on the platform. By default, the value is in kilobytes. You can designate different units by appending a single character modifier to the value: `M` or `m` for megabytes, `G` or `g` for gigabytes, or `T` or `t` for terabytes.

size = A positive integer equal to or greater than 1000 kilobytes and a multiple of the page size that specifies the size of the initial chunk of the dbspace. The value of *offset* plus the value of *size* cannot exceed the maximum chunk size. The maximum size of a chunk is equal to 2 147 483 647 pages multiplied by the page size. By default, the value is in kilobytes. You can designate different units by appending a single character modifier to the value: `M` or `m` for megabytes, `G` or `g` for gigabytes, or `T` or `t` for terabytes.

separators

Separate each value with a comma. Do not use blank spaces.

takes effect

After you edit your `onconfig` file and restart the SD secondary server.

Usage

The temporary dbspaces are created, or initialized if the dbspaces existed previously, when the SD secondary server starts. The temporary dbspaces are used for creating temporary tables. There must be at least one occurrence of the SDS_TEMPDBS configuration parameter in the `onconfig` file of the SD secondary server for the SD secondary server to start. You can specify up to 16 SD secondary temporary dbspaces in the `onconfig` file by using multiple occurrences of the SDS_TEMPDBS configuration parameter.

For each occurrence of the SDS_TEMPDBS configuration parameter in the `onconfig` file:

- The *dbsname* value must unique for each server and not shared with any other SD secondary server or the primary server.
- The combination of *dbspath*, *pagesize*, *offset*, and *size* must not cause any overlap with existing chunks or between temporary dbspaces spaces specified by the SDS_TEMPDBS configuration parameter.
- The *pagesize* value must be the same for each SDS_TEMPDBS configuration parameter value.

The following example shows two entries for the SDS_TEMPDBS configuration parameter:

```
SDS_TEMPDBS sds_space1,/dev/raw_dev1,2,0,60M
SDS_TEMPDBS sds_space2,/dev/raw_dev2,2,0,80M
```

If the primary server in a high-availability cluster fails and an SD secondary server takes over as the primary server, then the value set for the SDS_TEMPDBS configuration parameter on the SD secondary server is used for temporary dbspaces until the server is restarted. You should ensure that the value specified for the SDS_TEMPDBS configuration parameter on the SD secondary server is different than the value specified on the primary server. After the SD secondary server is restarted, the DBSPACETEMP configuration parameter is used.

SDS_TIMEOUT configuration parameter

Use the SDS_TIMEOUT configuration parameter to specify the amount of time in seconds that the primary server in a high-availability cluster will wait for a log-position acknowledgment to be sent from a shared disk (SD) secondary server.

onconfig.std value

SDS_TIMEOUT 20

if not present

`10`

values

2 - 2147483647

units

seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the SDS_TIMEOUT value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the SDS_TIMEOUT value in memory by running the onmode -wm command.

Usage

If no log-position acknowledgment is received from the SD secondary server in the specified amount of time, the primary server will disconnect from the SD secondary server and continue. After waiting for the number of seconds specified in the SDS_TIMEOUT configuration parameter setting, the primary server will start removing SD secondary servers, if page flushing has timed out while waiting for an SD secondary server.

SEC_APPLY_POLLTIME configuration parameter

Use the **SEC_APPLY_POLLTIME** configuration parameter to control how long log replay thread should poll for new work before yielding.

onconfig.std value

SEC_APPLY_POLLTIME 0

values

- Minimum value: (Default) 0
- Recommended value for smaller systems (between 1 to 8 CPUVPs): 0
- Recommended value for medium systems (between 8 and 16 CPUVPS): 10
- Recommended value for larger systems (> 16 CPUVPS): 1000
- Maximum value: 5000

units

micro seconds.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

In micro seconds, controls how long log replay thread should poll for new work before yielding. Use this parameter to reduce thread context switch overhead while replaying log records. It is recommended to configure poll threads to run on NET VP if SEC_APPLY_POLLTIME value > 0. For more information see, NETTYPE configuration parameter.

SEC_DR_BUFS configuration parameter

Use the **SEC_DR_BUFS** configuration parameter to control the number of replication buffers to be used for replicating log records to secondary server. Buffer size is same as LOGBUFF config value.

onconfig.std value

SEC_DR_BUFS 12

values

- Minimum value: (Default) 12
- Maximum value: 128
- Recommended value: Between 12 and 24

units

Number of replication buffers. Buffer size is same as LOGBUFF config value.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

SEC_DR_BUFS configuration parameter control the number of replication buffers to be used for replicating log records to secondary server. Buffer size is same as LOGBUFF config value.

SEC_LOGREC_MAXBUFS configuration parameter

Use the **SEC_LOGREC_MAXBUFS** configuration parameter to control the number of log buffers to be used for replaying log records at secondary server. Each log buffer is of size 16KB.

onconfig.std value

SEC_LOGREC_MAXBUFS 1000

values

- Minimum value: (Default) 5 times OFF_RECVRY_THREADS config parameter value
- Maximum value: Do not set to more than 2000 buffers
- Recommended value: 1000

units

Number of 16KB buffers

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

SEC_LOGREC_MAXBUFS configuration parameter control the number of log buffers to be used for replaying log records at secondary server.

SEC_NONBLOCKING_CKPT configuration parameter

Use the **SEC_NONBLOCKING_CKPT** configuration parameter to enable non-blocking checkpoint at HDR and RS secondary server.

onconfig.std value

SEC_NONBLOCKING_CKPT 0

values

- 1 - Enable non-blocking checkpoint at HDR and RS secondary server
- 0 – (Default) Disable non-blocking checkpoints

units

Not applicable

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

SEC_NONBLOCKING_CKPT configuration parameter controls the checkpoint behavior at HDR and RS secondary server.

When SEC_NONBLOCKING_CKPT configuration parameter is enabled, HDR secondary applies blocking or non-blocking checkpoint exactly like the Primary server.

When SEC_NONBLOCKING_CKPT configuration parameter is enabled, ensure that the size of the physical log is equal to at least 110% of the buffer pool size.

Related reference

PHYSFILE configuration parameter on page 144

SECURITY_LOCALCONNECTION configuration parameter

Use the SECURITY_LOCALCONNECTION configuration parameter to verify security on local connections by verifying that the ID of the local user who is running a program is the same ID of the user who is trying to access the database.

onconfig.std value

Not set.

values

`0` = No security checking occurs.

`1` = HCL OneDB™ checks whether the ID of the user who is running the program matches the ID of the user who is trying to connect to the database.

`2` = same as `1`, plus HCL OneDB™ retrieves the peer port number from the network API and verifies that the connection is coming from the client program. You can only specify two if your system has SOCTCP or IPCSTR network protocols.

takes effect

After you edit your `onconfig` file and restart the database server.

SEQ_CACHE_SIZE configuration parameter

Use the SEQ_CACHE_SIZE configuration parameter to specify the maximum number of sequence objects that are cached in memory.

onconfig.std value

SEQ_CACHE_SIZE 10

values

1 - 2147483647

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

When the maximum number of sequence objects are cached, the database server attempts to remove entries for any sequence objects that are no longer referenced.

SERVERNUM configuration parameter

The SERVERNUM configuration parameter specifies a relative location in shared memory.

onconfig.std value

SERVERNUM 0

values

0 - 255

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The value that you choose must be unique for each database server on your local computer. The value does not need to be unique on your network. Because the value `0` is included in the `onconfig.std` file, it is suggested that you choose a value other than `0` to avoid the inadvertent duplication of the SERVERNUM configuration parameter.

SESSION_LIMIT_LOCKS configuration parameter

The SESSION_LIMIT_LOCKS configuration parameter specifies the maximum number of locks available in a session. This limit does not apply to a user who holds administrative privileges, such as user **informix** or a DBSA user.

onconfig.std value

none

if not present

2147483647

values

500 - 2147483647

units

Number of locks in the internal lock table

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

For massively lock-intensive operations, administrators can set SESSION_LIMIT_LOCKS to reduces the risk of ordinary users in concurrent sessions depleting the lock resources of the database server.

The database server terminates a transaction that exceeds the limit of the number of locks, puts a message in the database server message log, and triggers the event alarm 21014.

Important:

In repeatable read isolation level, because each row in the active set requires a lock, be careful about setting too low a limit for locks on the server. Similarly, setting too small a lock limit can interfere with Enterprise Replication tasks or with cdr commands issued by non-DBSA users.

SESSION_LIMIT_LOGSPACE configuration parameter

The SESSION_LIMIT_LOGSPACE configuration parameter specifies the maximum amount of log space that a session can use for individual transactions. This limit does not apply to a user who holds administrative privileges, such as user **informix** or a DBSA user.

onconfig.std value

0 (off)

if not present

0 (off)

values

5120 - 2147483648

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The SESSION_LIMIT_LOGSPACE configuration parameter limits how much log space a session can use for each transaction, and can conserve system resources within a tenant-database environment.

The database server terminates a transaction that exceeds the log space limit, puts a message in the database server message log, and triggers the event alarm 21018.

The `session_limit_logspace` tenant database property set through the tenant create or tenant update SQL API command takes precedent over the SESSION_LIMIT_LOGSPACE configuration parameter setting.

SESSION_LIMIT_MEMORY configuration parameter

The SESSION_LIMIT_MEMORY configuration parameter specifies the maximum amount of memory that a session can allocate. This limit does not apply to a user who holds administrative privileges, such as user **informix** or a DBSA user.

onconfig.std value

0 (off)

if not present

0 (off)

values

20480 - 2147483648

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The SESSION_LIMIT_MEMORY configuration parameter limits how much memory a session can allocate, and can prevent individual sessions from monopolizing system resources.

The database server terminates a session that exceeds the memory limit, puts a message in the database server message log, and triggers the event alarm 21016.

The `session_limit_memory` tenant database property set through the tenant create or tenant update SQL API command takes precedent over the SESSION_LIMIT_MEMORY configuration parameter setting.

SESSION_LIMIT_TEMPSPACE configuration parameter

The SESSION_LIMIT_TEMPSPACE configuration parameter specifies the maximum amount of temporary table space that a session can allocate. This limit does not apply to a user who holds administrative privileges, such as user **informix** or a DBSA user.

onconfig.std value

0 (off)

if not present

0 (off)

values

20480 - 2147483648

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The SESSION_LIMIT_TEMPSPACE configuration parameter limits how much temporary table space a session can allocate, and can conserve system resources within a tenant-database environment.

The database server terminates a session that exceeds the space limit, puts a message in the database server message log, and triggers the event alarm 21017.

The `session_limit_tempspace` tenant database property set through the tenant create or tenant update SQL API command takes precedent over the SESSION_LIMIT_TEMPSPACE configuration parameter setting.

SESSION_LIMIT_TXN_TIME configuration parameter

The SESSION_LIMIT_TXN_TIME configuration parameter specifies the maximum amount of time that a transaction can run in a session. This limit does not apply to a user who holds administrative privileges, such as user **informix** or a DBSA user.

onconfig.std value

0 (off)

if not present

0 (off)

values

1 - 2147483647

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The SESSION_LIMIT_TXN_TIME configuration parameter limits how much time a transaction can run in a session, and can prevent individual session transactions from monopolizing the logical log.

The database server terminates a transaction that exceeds the time limit, puts a message in the database server message log, and triggers the event alarm 21019.

The `session_limit_txn_time` tenant database property set through the tenant create or tenant update SQL API command takes precedent over the SESSION_LIMIT_TXN_TIME configuration parameter setting.

SHMADD configuration parameter

Use the SHMADD configuration parameter to specify the size of the segments that are dynamically added to the virtual portion of shared memory.

onconfig.std value

Platform dependent

values

32-bit platforms: 1024 - 524288

64-bit platforms: 1024 - 4294967296

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The value of the SHMADD configuration parameter represents the size of the first set of segments that the database server adds to the virtual portion of shared memory when additional memory is needed. The size of the first virtual shared memory segment is set by the SHMVIRTSIZE configuration parameter. Set the values of the SHMVIRTSIZE and SHMADD configuration parameters so that a minimal number of segments are added during the normal operation of the database server. In general, more segments impair performance.

The value of the SHMADD configuration parameter represents the size of the segments that the database server adds to the virtual portion of shared memory when additional memory is needed. The size of the first virtual shared memory segment is set by the SHMVIRTSIZE configuration parameter. Set the values of the SHMVIRTSIZE and SHMADD configuration parameters so that a minimal number of segments are added during the normal operation of the database server. In general, more segments impair performance. It is more efficient to add memory in large segments, but wasteful if the added memory is not used. Also, the operating system might require you to add memory in a few large segments rather than many small segments.

The maximum number of HCL OneDB™ shared memory segments is 1024. Many shared memory segments might be required if the SHMADD value is low or the database server has unexpectedly large amounts of activity or memory use. To prevent the database server from reaching the maximum number of shared memory segments, the size of virtual segments that are added dynamically by the server doubles every 16 virtual segments. It is more efficient to add memory in large segments, but wasteful if the added memory is not used. Also, the operating system might require you to add memory in a few large segments rather than many small segments.

The following table contains recommendations for setting the initial value of SHMADD.

Table 71. Recommended SHMADD values

Amount of physical memory	Recommended SHMADD value
Less than 256 MB	8192
256 - 512 MB	16,384
Greater than 512 MB	32,768

You can view information about virtual memory segments by running the onstat -g seg command.

SHMBASE configuration parameter

Use the SHMBASE configuration parameter to specifiy the base address where shared memory is attached to the memory space of a virtual processor.

onconfig.std value

Platform dependent

values

Positive integers

units

Address

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The addresses of the shared-memory segments start at the SHMBASE value and grow until the upper-bound limit, which is platform specific.

Do not change the value of SHMBASE. The `onconfig.std` value for SHMBASE depends on the platform and whether the processor is 32-bit or 64-bit. For information on which SHMBASE value to use, see the machine notes.

SHMNOACCESS configuration parameter

The SHMNOACCESS configuration parameter specifies a virtual memory address range to not use to attach shared memory.

onconfig.std values

On UNIX™: None

On Windows™: #SHMNOACCESS 0x70000000-0x7FFFFFFF, and this value is commented out in the onconfig.std template file.

values

1 - 10 address ranges

separators

Comma

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The SHMNOACCESS configuration parameter is used to avoid specific range process addresses, which in turn avoids conflicts with operating system libraries.

Each address in each range must start in hexadecimal format. Each address in a range must be separated by a hyphen and each range must be separated by a comma, as the following example shows:

```
SHMNOACCESS 0x70000000-0x75000000,
0x7A000000-0x80000000
```

SHMTOTAL configuration parameter

Use the SHMTOTAL configuration parameter to specify the total amount of shared memory (resident, virtual, communications, and virtual extension portions) to be used by the database server for all memory allocations. The `onconfig.std` value of `0` implies that no limit on memory allocation is stipulated.

onconfig.std value

SHMTOTAL 0

values

`0` = (no specific limit) or any integer greater than or equal to 1

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

You can use the SHMTOTAL configuration parameter to limit the demand for memory that the database server can place on your system. However, applications might fail if the database server requires more memory than the limit imposed by SHMTOTAL. When this situation occurs, the database server writes the following message in the message log:

```
size of resident + virtual segments xx + yy > zz total allowed by
configuration parameter SHMTOTAL
```

This message includes the following values.

Value

Description

xx

Current® size of resident segments

yy

Current® size of virtual segments

zz

Total shared memory required

If you enabled the LOW_MEMORY_MGR configuration parameter and are configuring the server to use a percentage of the SHMTOTAL configuration parameter value for automatic low memory management start and stop thresholds, the SHMTOTAL configuration parameter must not be set to 0 (unilmited).

Attention: Changing the value of the SHMTOTAL configuration parameter value can cause the configuration of automatic low memory management to become invalid, forcing the database server to use the default settings.

UNIX Only:

Set the operating-system parameters for maximum shared-memory segment size, typically SHMMAX, SHMSIZE, or SHMALL, to the total size that your database server configuration requires. For information about the amount of shared memory that your operating system allows, see the machine notes.

If you have more physical memory than the value specified in the machine notes, and the memory is to be used by HCL OneDB™, you can increase the value of the SHMALL parameter to as much 90 percent of the physical memory that is specified for your computer. It is recommended that you do not meet or exceed the available RAM.

SHMVIRT_ALLOCSEG configuration parameter

Use the SHMVIRT_ALLOCSEG configuration parameter to specify a threshold at which HCL OneDB™ should allocate a new shared memory segment and the level of the event alarm activated if the server cannot allocate the new memory segment.

onconfig.std value

SHMVIRT_ALLOCSEG 0,3

values

A numeric value optionally followed by a comma and another numeric value.

threshold = A number that indicates when the database server should add a shared memory segment:

- 0 = Default. The database server allocated shared memory segments when needed.
- .40 - .99 = The percentage of memory used before a segment is added.
- 256 - 10000000 = The number of kilobytes remaining before a segment is added.

alarm_level: Optional. An integer value from 1 to 5 that specifies the level of the event alarm to raise: 1 = Not noteworthy, 2 = Information, 3 = Attention (Default), 4 = Emergency, 5 = Fatal. The event alarm has a class ID of 24 and an event ID of 24003.

separator

Separate the values with a comma.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

Set the SHMVIRT_ALLOGSEG configuration parameter to proactively add shared memory segments instead of waiting until the database server automatically adds shared memory segments.

The event alarm repeats every thirty minutes if a new memory segment cannot be allocated.

SHMVIRTSIZE configuration parameter

Use the SHMVIRTSIZE configuration parameter to specify the initial size of a virtual shared-memory segment.

onconfig.std value

Platform dependent

if not present

If SHMADD is present: the value of the SHMADD configuration parameter.

If SHMADD is not present: 8192.

values

32-bit platforms: Positive integer with a maximum value of 2 GB

64-bit platforms: Positive integer with a maximum value of 4 TB

The maximum value might be less on some platforms due to operating-system limitations. For the actual maximum value for your UNIX™ platform, see the machine notes.

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

To determine the appropriate value for the SHMVIRTSIZE configuration parameter, use the following algorithm to determine the size of the virtual portion of shared memory:

```
shmvirtsize = fixed overhead + ((stack size + heap) * number of users)
```

Variable	Value to use
fixed overhead	This includes the size of the AIO vectors, sort memory, dbspace backup buffers, dictionary size, size of stored-procedure cache, histogram pool, other pools, and other overhead. To obtain an estimate of the fixed overhead, start the database server and see how many additional memory segments are allocated, if any. The number of users that you have on the system when you start the server, impacts the allocation of memory segments. When you start the server: • If the number of users is typical for your environment, then add the size of the memory segments to the current value for the SHMVIRTSIZE configuration parameter and restart the server. • If the number of users is far less than what is typical for your environment, you must calculate the appropriate overhead value to use for the memory segments. You can determine how many memory segments each user consumes by dividing the number of additional memory segments that are allocated when you started the server by the number of users that you had on the server then. Multiply the value for the memory segments for each user by the number of users that you typically have on the system. Add this calculated value for the memory segments to the current value for SHMVIRTSIZE configuration parameter and restart the server.
stack size	On 32-bit systems, use 32 KB for the stack size. Typically on 64-bit systems, you use 64 KB for the stack size. However, some 64-bit systems use a different value.
heap	Use 30 KB per user.
number of users	Use the maximum number of concurrent user sessions that you anticipate on the server.

If possible, create a virtual portion of shared memory of a size that is more than you require for your daily processing.

Use the onstat -g seg command to determine peak usage and lower the value of the SHMVIRTSIZE configuration parameter accordingly.

SINGLE_CPU_VP configuration parameter

The SINGLE_CPU_VP configuration parameter specifies whether or not the database server is running with only one CPU virtual processor.

onconfig.std value

SINGLE_CPU_VP 0

values

`0` = running with multiple CPU VPs

`1` = running with one CPU VP

takes effect

When the database server is shut down and restarted

Usage

Disable the SINGLE_CPU_VP configuration parameter by setting it to 0 if you want the number of CPU VPs to be automatically increased when the database server starts.

Setting SINGLE_CPU_VP to nonzero allows the database server to use optimized code based on the knowledge that only one CPU virtual processor is running. It enables the database server to bypass many of the mutex calls that it must use when it runs multiple CPU virtual processors.

It is strongly recommended that you set this parameter when the database server will run only one CPU virtual processor. Depending on the application and workload, setting this parameter can improve performance by up to 10 percent.

If you set SINGLE_CPU_VP to nonzero and try to add a CPU virtual processor, you receive one of the following messages:

```
onmode: failed when trying to change the number of classname VPs by n.
onmode: failed when trying to change the number of cpu virtual processors by n.
```

If you set SINGLE_CPU_VP to nonzero and then attempt to bring up the database server with VPCLASS **cpu,** *num* set to a value greater than 1, you receive the following error message, and the database server initialization fails:

```
Cannot have SINGLE_CPU_VP non-zero and CPU VPs greater than 1.
```

VPCLASS Values and the SINGLE_CPU_VP Configuration Parameter

HCL OneDB™ treats user-defined virtual-processor classes as if they were CPU virtual processors. If you set the *SINGLE_CPU_VP* configuration parameter to a nonzero value, you cannot create any user-defined virtual-processor classes.

Using a user-defined VPCLASS

If you set this configuration parameter to a nonzero value and then attempt to bring up the database server with a user-defined VPCLASS, you receive the following error message, and the database server initialization fails:

```
oninit: Cannot have SINGLE_CPU_VP non-zero and user-defined VP classes
```

Using the *cpu* VPCLASS

If you set this configuration parameter to a nonzero value and then attempt to bring up the database server with the VPCLASS *cpu* value for *num* set to a value greater than 1, you receive the following error message, and the database server initialization fails:

```
Cannot have SINGLE_CPU_VP non-zero and CPU VPs greater than 1.
```

SMX_COMPRESS configuration parameter

Use the SMX_COMPRESS configuration parameter to specify the level of compression that the database server uses before sending data from the source database server to the target database server.

Network compression saves network bandwidth over slow links but uses more CPU to compress and decompress the data. The SMX_COMPRESS configuration parameter values of the two servers are compared and changed to the higher compression values.

Note: From version 2.0.0.0 onwards, Enterprise Replication will use SMX connection for communicating with the peer servers.

onconfig.std value

SMX_COMPRESS 0

values

`-1` = The source database server never compresses the data, regardless of whether or not the target site uses compression.

`0` = The source database server compresses the data only if the target database server expects compressed data.

`1` = The database server performs a minimum amount of compression.

`9` = The database server performs the maximum possible compression.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

SMX_NUMPIPES configuration parameter

The SMX_NUMPIPES configuration parameter sets the number of pipes for server multiplexer group (SMX) connections.

Note: From version 2.0.0.0 onwards, Enterprise Replication will use SMX connection for communicating with the peer servers.

onconfig.std value

SMX_NUMPIPES 1

values

1 - 32767 = The number of network pipes for SMX connections.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

High-availability clusters and parallel sharded queries use SMX connections. If the lag time between servers is too long, increase the number of SMX pipes.

SMX_PING_INTERVAL configuration parameter

Use the SMX_PING_INTERVAL configuration parameter to specify the number of seconds in a timeout interval, where a secondary server waits for activity from the primary server in a Server Multiplexer Group (SMX) connection.

Note: From version 2.0.0.0 onwards, Enterprise Replication will use SMX connection for communicating with the peer servers.

onconfig.std value

SMX_PING_INTERVAL 10

values

`0` = Wait indefinitely.

A positive integer between `1` and `60`, inclusive. = The number of seconds in the timeout interval.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

After you run the SQL administration API task() or admin() function with the `"onmode","-wf SMX_PING_INTERVAL=value"` or `"onmode","-wm SMX_PING_INTERVAL=value"` argument.

Usage

If the secondary server does not receive any message during the length of time that is specified by the SMX_PING_INTERVAL configuration parameter and after the number of intervals that are specified by the SMX_PING_RETRY configuration parameter, the secondary server prints an error message to the `online.log` and closes the SMX connection. If an SMX timeout message is in the `online.log`, you can increase the SMX_PING_INTERVAL value, the SMX_PING_RETRY value, or both of these values.

This configuration parameter applies only to secondary servers. If you set SMX_PING_INTERVAL on the primary server, it becomes effective if the primary server becomes a secondary server.

Example

If the `onconfig` file of a secondary server in a high-availability cluster has the following entries, the secondary server waits a total of 180 seconds for activity from the primary server. If there is no activity from the primary server during those 180 seconds, the secondary server closes the SMX connection and writes an error message to the online log.

```
SMX_PING_INTERVAL 30
SMX_PING_RETRY 6
```

SMX_PING_RETRY configuration parameter

Use the SMX_PING_RETRY configuration parameter to specify the maximum number of times that a secondary server repeats the timeout interval that is specified by the SMX_PING_INTERVAL configuration parameter if a response from the primary server is not received. If the maximum number is reached without a response, the secondary server prints an error message in the `online.log` and closes the Server Multiplexer Group (SMX) connection.

Note: From version 2.0.0.0 onwards, Enterprise Replication will use SMX connection for communicating with the peer servers.

onconfig.std value

SMX_PING_RETRY 6

values

Any positive integer = The maximum number of times to repeat the timeout interval.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

After you run the SQL administration API task() or admin() function with the After you run the SQL administration API task() or admin() function with the `"onmode","-wf SMX_PING_RETRY=value"` or `"onmode","-wm SMX_PING_RETRY=value"` argument.

Usage

If the secondary server does not receive any message during the length of time that is specified by the SMX_PING_INTERVAL configuration parameter and after the number of intervals that are specified by the SMX_PING_RETRY configuration parameter, the secondary server prints an error message to the `online.log` and closes the SMX connection. If an SMX timeout message is in the `online.log`, you can increase the SMX_PING_INTERVAL value, the SMX_PING_RETRY value, or both of these values.

This configuration parameter applies only to secondary servers. If you set SMX_PING_RETRY on the primary server, it becomes effective if the primary server becomes a secondary server.

Example

If the `onconfig` file of a secondary server in a high-availability cluster has the following entries, the secondary server waits a total of 60 seconds for activity from the primary server. If there is no activity from the primary server during those 60 seconds, the secondary server closes the SMX connection and writes an error message to the online log.

```
SMX_PING_INTERVAL 12
SMX_PING_RETRY 5
```

SP_AUTOEXPAND configuration parameter

Use the SP_AUTOEXPAND configuration parameter to enable or disable the automatic creation or extension of chunks.

onconfig.std value

SP_AUTOEXPAND 1

values

0 = The automatic creation or extension of chunks is not enabled.

1 = The automatic creation or extension of chunks is enabled.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

When the SP_AUTOEXPAND configuration parameter is enabled and a storage container such as a dbspace has a defined create size or extend size that is not zero, the container is auto-expandable.

SP_THRESHOLD configuration parameter

Use the SP_THRESHOLD configuration parameter to define the minimum amount of free kilobytes that can exist in a storage space before HCL OneDB™ automatically runs a task to expand the space, either by extending an existing chunk in the space or by adding a new chunk.

onconfig.std value

SP_THRESHOLD 0

values

`0` = No threshold. The trigger that runs the storage space monitoring (**mon_low_storage**) task for adding space when space is below the threshold is disabled.

`1 - 50` = A threshold that is a percentage of free kilobytes in a storage space.

If the value is 50 or below, HCL OneDB™ interprets the value as a percentage (for example, 10 = 10 percent and 2.84 = 2.84 percent).

`1000` to the maximum size of a chunk = A threshold that is either 1000 kilobytes or the maximum size of the chunk on the current platform.

If the value is 1000 or higher, HCL OneDB™ interprets the value as a specific number of kilobytes.

Values 50 - 1000 are not valid.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

When you set the SP_THRESHOLD configuration parameter to a valid value that is greater than 0, the built-in Scheduler task, **mon_low_storage**, runs automatically when the free space in a dbspace, temporary dbspace, sbspace, temporary sbspace, or blobspace falls below the threshold.

Suppose the value of the SP_THRESHOLD configuration parameter value is 5.5, which the server interprets as 5.5 percent. If a space runs low on free pages, and the free space percentage falls below 5.5 percent and remains below that level until the **mon_low_storage** task runs next, that task will attempt to expand the space. If the SP_THRESHOLD configuration parameter is set to 50000 and a space has fewer than 50000 free kilobytes, that space will be expanded the next time **mon_low_storage** task runs.

A value of 0 turns off the **mon_low_storage** task, and prevents the server from extending any space. However, a value of 0 does not affect the ability of the server to extend a space when all free pages are depleted and more are needed.

The value specified in the SP_THRESHOLD configuration parameter applies to all spaces belonging to the server.

SP_WAITTIME configuration parameter

Use the SP_WAITTIME configuration parameter to specify the maximum number of seconds that a thread waits for a dbspace, temporary dbspace, plogspace, sbspace, temporary sbspace, or blobspace space to expand before returning an out-of-space error. Use the SP_WAITTIME configuration parameter to specify the maximum number of seconds that a thread waits for a dbspace, temporary dbspace, sbspace, temporary sbspace, or blobspace space to expand before returning an out-of-space error.

onconfig.std value

SP_WAITTIME 30

values

0 - 2147483647

units

seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The time that the server uses to automatically add or expand a chunk can vary widely, depending on various factors such as the size of the chunk, the speed of the associated disk drives, and the load on the system. When HCL OneDB™ automatically adds or expands a chunk to prevent free space from falling below the threshold specified by the SP_THRESHOLD configuration parameter, HCL OneDB™ forces threads that need the space to wait until it is available. You can change the value of the SP_WAITTIME configuration parameter if you want to change the maximum amount of time that the thread will wait for more space.

A thread will wait for a storage space to expand only if the storage pool contains entries. A thread will not wait if the storage pool is empty.

SQL_LOGICAL_CHAR configuration parameter

Use the SQL_LOGICAL_CHAR configuration parameter to enable or disable the expansion of size specifications in declarations of built-in character data types.

onconfig.std value

SQL_LOGICAL_CHAR OFF (= interpret size specifications in units of bytes)

values

`OFF` = No expansion of declared sizes.

`1` = No expansion of declared sizes.

`2` = Use `2` as the expansion factor for declared sizes.

`3` = Use `3` as the expansion factor for declared sizes.

`4` = Use `4` as the expansion factor for declared sizes.

`ON` = Use *M* as the expansion factor, where *M* is the maximum length in bytes that any logical character requires in the code set of the current database. Depending on the **DB_LOCALE** setting, *M* has an integer range from `1` (in single-byte locales) up to `4`.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

For applications that are developed in single-byte locales, but deployed in multibyte locales, this feature can reduce the risk of multibyte logical characters being truncated during data entry operations.

In a multibyte code set, such as **UTF-8** or the multibyte code sets for some East Asian languages, a single logical character can require more than one byte of storage. The setting of this parameter can instruct the SQL parser to apply logical-character semantics to declarations of these built-in character data types:

- BSON
- CHAR
- CHARACTER
- CHARACTER VARYING
- JSON
- LVARCHAR
- NCHAR
- NVARCHAR
- VARCHAR
- DISTINCT types that declare any of these data types as their base types
- ROW types (named and unnamed) that include fields of these data types
- Collection types (LIST, MULTISET, or SET) that include these types as elements.

The setting that you specify for this parameter must be one of the following values:

Whether the SQL_LOGICAL_CHAR configuration parameter is set to enable or disable the expansion of declared storage sizes, its setting specifies how data type declarations are interpreted for all sessions of the HCL OneDB™ instance.

Automatic Resizing of the Expansion Factor

When SQL_LOGICAL_CHAR is set to a valid digit, and the current session creates a database, HCL OneDB™ compares the SQL_LOGICAL_CHAR value with the maximum number of bytes that any logical character will use for the code set of the database.

If the SQL_LOGICAL_CHAR setting is greater than that maximum number of bytes, the database uses the maximum value for the locale as the new expansion factor, overriding what the configuration file specifies. The SQL_LOGICAL_CHAR setting in the configuration file remains unchanged, and continues to act as the default expansion factor for other user databases.

Similarly, if the SQL_LOGICAL_CHAR value for a session is automatically reset to a digit, as described above, but the same session subsequently connects to another database whose locale uses a code set in which a logical character requires a larger storage size than the current expansion factor, HCL OneDB™ uses the maximum number of bytes for the new code set as the new expansion factor while the user session is connected to that database, rather than using the current setting of SQL_LOGICAL_CHAR.

Automatic resetting of the expansion factor to match the largest logical character size in the code set that **DB_LOCALE** specifies at connection time also occurs when SQL_LOGICAL_CHAR is set to `ON`, but the effects of the `ON` setting are not identical to the database server behavior when SQL_LOGICAL_CHAR is set to a digit (1, 2, 3, or 4) in two ways:

- The expansion factor can be automatically reset to a smaller value if `ON` is the SQL_LOGICAL_CHAR setting.
- There is no difference between `SQL_LOGICAL_CHAR = 4` and `SQL_LOGICAL_CHAR = ON`.

You must set SQL_LOGICAL_CHAR to `ON`, rather than to a digit, if you want a smaller expansion factor when the current session connects to a database whose largest logical character in the **DB_LOCALE** code set requires a smaller number of bytes than the current SQL_LOGICAL_CHAR setting. The effective expansion factor will always be less than or equal to the maximum character size for a locale.

SQLTRACE configuration parameter

Use the SQLTRACE parameter to control the startup environment of SQL tracing.

onconfig.std value

#SQLTRACE level=low,ntraces=1000,size=2,mode=global

On UNIX™: Not set. SQL tracing is not enabled.

On Windows™: #SQLTRACE level=low,ntraces=1000,size=2,mode=global

values

See the Usage section.

takes effect

After you edit your `onconfig` file and restart the database server.

After you run the SQL administration API task() or admin() function with the set sql tracing argument.

Usage

Remove the # symbol from the `onconfig` value to retain basic information, up to 2 KB in size, about the last 1000 SQL statements that were run by any user. You can customize the scope of the SQL tracing information by adjusting the field values of the SQLTRACE configuration parameter.

Syntax for the SQLTRACE configuration parameter

SQLTRACE [**level=** { **low** | **medium** | **high** | **off** } ,] [**ntraces=** { **1000** | *number_traces* } ,] [**size=** { **2** | *buffer_size* } ,] [**mode=** { **global** | **user** }]

Table 72. Options for the SQLTRACE configuration parameter value

Field	Values
level	Amount of information traced: • `Low` = Default. Captures statement statistics, statement text, and statement iterators. • `Medium` = Captures all of the information included in low-level tracing, plus table names, the database name, and stored procedure stacks. • `High` = Captures all of the information included in medium-level tracing, plus host variables. • `Off` = Specifies no SQL tracing.
ntraces	The *number_traces* value is the number of SQL statements to trace before reusing the resources. Default is 1000. The range is 500 - 2147483647.
size	The *buffer_size* value is the maximum size of variable length data to be stored, in KB. Default is 2. The range is 1 -100. If this buffer size is exceeded, the database server discards saved data.
mode	Scope of tracing performed: • `Global` = Default. All users. • `User` = Users who have tracing enabled by an SQL administration API task() or admin() function. Specify this mode if you want to get a sample of the SQL that a small set of users is running.

The onstat -g his command displays SQL trace information.

STACKSIZE configuration parameter

Use the STACKSIZE configuration parameter to specify the stack size for the database server user threads.

onconfig.std value

STACKSIZE 32 for 32-bit database servers

STACKSIZE 64 for 64-bit database servers

values

32 through limit determined by the database server configuration and the amount of memory available

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The value of STACKSIZE does not have an upper limit, but setting a value that is too large wastes virtual memory space and can cause swap-space problems.

For 32-bit platforms, the default STACKSIZE value of 32 kilobytes is sufficient for nonrecursive database activity. For 64-bit platforms, the recommended STACKSIZE value is 64 kilobytes. When the database server performs recursive database tasks, as in some SPL routines, for example, it checks for the possibility of stack-size overflow and automatically expands the stack.

User threads execute user-defined routines. To increase the stack size for a particular routine, use the **stack** modifier on the CREATE FUNCTION statement.

Warning: Setting the value of STACKSIZE too low can cause stack overflow, the result of which is undefined but usually undesirable.

STAGEBLOB configuration parameter

Use this parameter only if you are storing TEXT or BYTE data on optical storage with the Optical Subsystem. This parameter has no effect on ordinary blobspaces or sbspaces.

onconfig.std value

Not set.

values

Up to 128 bytes. STAGEBLOB must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or **$** characters.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The STAGEBLOB configuration parameter is the blobspace name for the area where the Optical Subsystem stages TEXT and BYTE data that is destined for storage on optical disk.

STATCHANGE configuration parameter

Use the STATCHANGE configuration parameter to specify a positive integer for a global percentage of a change threshold for the server to use to determine if distribution statistics qualify for an update when the automatic mode for UPDATE STATISTICS operations is enabled.

onconfig.std value

STATCHANGE 10

values

0 - 100

units

percentage of a change threshold

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The database server uses the value of the STATCHANGE configuration parameter when the AUTO_STAT_MODE configuration parameter, the AUTO_STAT_MODE session environment variable, or the AUTO keyword of the UPDATE STATISTICS statement has enabled the automatic mode for UPDATE STATISTICS operations.

The STATCHANGE setting specifies a change threshold for the database server to use to determine if distribution statistics qualify for an update when the automatic mode for UPDATE STATISTICS operations is enabled. When this mode is enabled, the UPDATE STATISTICS statement compares the STATCHANGE setting with the percentage of rows that have changed in each table or fragment since the current data distributions were calculated, and selectively updates only the missing or stale distribution statistics for each table or fragment within the scope of the UPDATE STATISTICS statement.

STMT_CACHE configuration parameter

Use the STMT_CACHE configuration parameter to determine whether the database server uses the SQL statement cache.

onconfig.std value

`STMT_CACHE 0`

values

`0` = SQL statement cache not used (equivalent to onmode -e `OFF`).

`1` = SQL statement cache enabled, but user sessions do not use the cache. Users use the cache only if they set the environment variable **STMT_CACHE** to `1` or execute the SQL statement SET STATEMENT CACHE ON.

`2` = SQL statement cache turned on. All statements are cached. To turn off statement caching, set the environment variable **STMT_CACHE** to `0` or execute the SQL statement SET STATEMENT CACHE OFF.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

You can enable the SQL statement cache in one of two modes:

- Always use the SQL statement cache unless a user explicitly specifies not to use it. Set the STMT_CACHE configuration parameter to `2` or onmode -e `ON`.
- Use the SQL statement cache only when a user explicitly specifies to use it. Set the STMT_CACHE configuration parameter to `1` or onmode -e `ENABLE`.

STMT_CACHE_HITS configuration parameter

Use the STMT_CACHE_HITS configuration parameter to specify the number of hits (references) to a statement before it is fully inserted in the SQL statement cache.

onconfig.std value

STMT_CACHE_HITS 0

values

`0` = Fully insert all qualified statements in the SQL statement cache.

`>0` = The first time a user issues a unique statement, the database server inserts a *key-only* entry in the cache that identifies the statement. Subsequent identical statements increment the hit count of the *key-only* cache entry. When the hit count of the *key-only* cache entry reaches the specified number of hits, the database server fully inserts the statement in the cache. Set *hits* to `1` or more to exclude ad hoc queries from entering the cache.

units

Integer

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

STMT_CACHE_NOLIMIT configuration parameter

Use the STMT_CACHE_NOLIMIT configuration parameter to control whether to insert qualified statements into the SQL statement cache.

onconfig.std value

STMT_CACHE_NOLIMIT 0

if not present

1

values

`0` = Prevents statements from being inserted in the cache. The cache can grow beyond the size limit if most of the statements in the cache are currently in use, because the cache cleaning cannot catch up with the insert rate. If you are concerned about memory usage, turn off STMT_CACHE_NOLIMIT to prevent the database server from allocating a large amount of memory for the cache.

`1` = Always insert statements in the SQL statement cache regardless of the cache size.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

STMT_CACHE_NUMPOOL configuration parameter

Use the STMT_CACHE_NUMPOOL configuration parameter to specify the number of memory pools for the SQL statement cache. To obtain information about these memory pools, use onstat -g ssc pool.

Because the database server does not insert all statements that allocate memory from the memory pools in the cache, the cache size might be smaller than the total size of the memory pools.

onconfig.std value

STMT_CACHE_NUMPOOL 1

values

1 - 256

units

Positive integer

takes effect

After you edit your `onconfig` file and restart the database server.

STMT_CACHE_SIZE configuration parameter

Use the STMT_CACHE_SIZE configuration parameter to specify the size of the SQL statement caches in kilobytes. The new cache size takes effect the next time a statement is added to a cache.

onconfig.std value

STMT_CACHE_SIZE 512

values

Positive integer

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

STOP_APPLY configuration parameter

Use the STOP_APPLY configuration parameter to stop an RS secondary server from applying log files received from the primary server.

onconfig.std value

STOP_APPLY 0

default value

0

values

0 = Apply logs

1 = Stop applying logs immediately

YYYY:*MM*:*DD*-*hh*:*mm*:*ss* = Stop the log apply at a specified time, where:

- *YYYY* = Year
- *MM* = Month
- *DD* = Day
- *hh* = Hour (24-hour notation)
- *mm* = Minute
- *ss* = Second

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Stopping the application of log files allows you to recover quickly from erroneous database modifications by restoring the data from the RS secondary server. You can configure the server to either stop the application of logs immediately, or at a specified point in time. When setting the value of STOP_APPLY you must also set LOG_STAGING_DIR. If STOP_APPLY is configured and LOG_STAGING_DIR is not set to a valid and secure directory, the server cannot be initialized.

Log files are stored in binary format in a directory specified by the LOG_STAGING_DIR configuration parameter. You must specify a valid and secure location for the log files.

To see information about the data being sent to the log-staging directory set for a RS secondary server, run the onstat -g rss verbose command on the RS secondary server.

If the write to the staging file fails, the RS secondary server raises event alarm 40007.

The time value specified for the STOP_APPLY configuration parameter is assumed to be in the same timezone as the RS secondary server.

The dbexport utility cannot support write operations on an updatable secondary server unless the STOP_APPLY parameter is set. (Besides STOP_APPLY, the UPDATABLE_SECONDARY and USELASTCOMMITTED configuration parameters must also be set to enable write operations by dbexport on a secondary data replication server.)

If a remote stand-alone secondary (RSS) server has its STOP_APPLY configuration parameter set to a value other than 0, that server cannot use cluster transaction coordination.

STORAGE_FULL_ALARM configuration parameter

Use the STORAGE_FULL_ALARM configuration parameter to configure the frequency and severity of messages and alarms when storage spaces become full.

onconfig.std value

STORAGE_FULL_ALARM 600,3

values

seconds = `0` (off) or a positive integer indicating the number of seconds between notifications.

severity_level = `0` (no alarms) or 1 - 5

units

seconds,*severity_level*

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

When a storage space, such as a dbspace, sbspace, blobspace, or tblspace, or a partition becomes full, an alarm is raised and a message is sent to the online message log. You can specify the number of seconds between notifications with the first value of this parameter. You can specify the lowest severity for event alarms to be returned. Setting a specific severity prevents events that have a lower severity from being raised. But events that have the same or greater severity as the severity specified are raised. You can prevent alarms when storage spaces become full by setting this parameter to `0`.

Regardless of the value of STORAGE_FULL_ALARM, messages are sent to the online message log when storage spaces or partitions become full.

SYSALARMPROGRAM configuration parameter

Use the SYSALARMPROGRAM configuration parameter to specify the full path name of the `evidence.sh` script. The database server executes `evidence.sh` when a database server failure occurs. You can use the output from the `evidence.sh` script to diagnose the cause of a database server failure.

onconfig.std value

On UNIX™: `$ONEDB_HOME/etc/evidence.sh`
On Windows™: Not set. (Commented out.) Listed as `$ONEDB_HOME\etc\evidence.bat`

values

pathname = Full path name of the `evidence.sh` script.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

On Windows™, you must enable command extensions for `evidence.bat` to successfully complete. You can enable and disable the extensions for the Command Prompt you are working in by issuing the following commands:

- Enable: cmd /x
- Disable: cmd /y

You can also enable and disable command extensions from the Windows™ XP registry:

Table 73. Enabling command extensions from the Windows™ registry

Attribute	Value
Hive	HKEY_CURRENT_USER
Key	Software\Microsoft\Command Processor
Name	EnableExtensions
Type	REG_DWORD
Values	0 (disable), 1 (enable)

SYSSBSPACENAME configuration parameter

Use the SYSSBSPACENAME configuration parameter to specify the name of the sbspace in which the database server stores fragment-level data-distribution statistics, which the **syfragsdist** system catalog table stores as BLOB objects in its **encsdist** column. Also use SYSSBSPACENAME to specify the name of the sbspace in which the database server stores statistics that the UPDATE STATISTICS statement collects for certain user-defined data types.

onconfig.std value

Not set.

if not present

`0`

values

Up to 128 bytes. SYSSBSPACENAME must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or **$** characters.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

refer to

- Updating statistics, in the chapter on individual query performance in your *HCL OneDB™ Performance Guide*
- Sbspace characteristics, in the chapter on configuration effects on I/O in your *HCL OneDB™ Performance Guide*

- Writing user-defined statistics, in the performance chapter in *HCL OneDB™ User-Defined Routines and Data Types Developer's Guide*
- Providing statistics data for a column, in the *HCL OneDB™ DataBlade® API Programmer's Guide*

Usage

To support fragment level statistics, you must specify the name of an sbspace as the SYSSBSPACENAME setting, and you must allocate that sbspace (by using the onspaces utility, as described below. For any table whose STATLEVEL attribute is set to FRAGMENT, the database server returns an error if SYSSBSPACENAME is not set, or if the sbspace to which SYSSBSPACENAME is set was not properly allocated).

For the distribution statistics of a column in a fragmented table, you can estimate how many bytes of storage capacity the sbspace requires by this formula:

```
nfrags * 1.25 * ((10000 / resolution) * ((2 * column_width) + 6))
```

Here 1.25 approximates the number of overflow bins. The formula also includes these variables:

- `column_width` is the width in bytes of the column that the UPDATE STATISTICS statement specifies.
- `nfrags` is the number of fragments of the table.
- `resolution` is the *percent* value in the resolution clause of the UPDATE STATISTICS statement that calculates the distribution.

The `resolution` is also what the dbschema -hd *table* command displays for the column distribution statistics.

SYSSBSPACENAME also specifies the name of the sbspace in which the database server stores statistics that the UPDATE STATISTICS statement collects for certain user-defined data types. Normally, the database server stores statistics in the **sysdistrib** system catalog table.

Do not confuse the SYSSBSPACENAME configuration parameter with the SBSPACENAME configuration parameter .

Because the data distributions for user-defined data types can be large, you have the option to store them in an sbspace instead of in the **sysdistrib** system catalog table. If you store the data distributions in an sbspace, use DataBlade® API or functions to examine the statistics.

Even though you specify an sbspace with the SYSSBSPACENAME parameter, you must create the sbspace with the -c -S option of the onspaces utility before you can use it. The database server validates the name of this sbspace when one of the following occurs:

- The database server attempts to write data distributions of the multirepresentational type to SYSSBSPACENAME when it executes the UPDATE STATISTICS statement with the MEDIUM or HIGH keywords.
- The database server attempts to delete data distributions of the multirepresentational type to SYSSBSPACENAME when it executes the UPDATE STATISTICS statement with the DROP DISTRIBUTIONS keywords.

If SBSSPACENAME is not set, or if storage is not allocated to that sbspace, the database server might not be able to store the distribution statistics, so that the UPDATE STATISTICS operation fails with error -9814.

Although you can store smart large objects in the sbspace specified in SYSSBSPACENAME, keeping the distribution statistics and smart large objects in separate sbspaces is recommended, because:

- You avoid disk contention when queries are accessing smart large objects, and the query optimizer is using the distributions to determine a query plan.
- Disk space takes longer to fill up when each sbspace is used for a different purpose.

TBLSPACE_STATS configuration parameter

Use the TBLSPACE_STATS configuration parameter to turn on and off the collection of tblspace statistics. Use the onstat -g ppf command to list tblspace statistics.

onconfig.std value

TBLSPACE_STATS 1

values

`0` = Turn off the collection of tblspace statistics. The onstat -g ppf command displays `partition profiles disabled`.

`1` = Turn on the collection of tblspace statistics.

units

Integer

takes effect

After you edit your `onconfig` file and restart the database server.

TBLTBLFIRST configuration parameter

Use the TBLTBLFIRST configuration parameter if you want to specify the first extent size of tblspace **tblspace** in the root dbspace. Set this parameter if you do not want the database server to automatically manage the extent size.

onconfig.std value

TBLTBLFIRST 0

values

From the equivalent of 250 pages specified in kilobytes to the size of the first chunk minus the space needed for any system objects.

units

Kilobytes in multiples of page size

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

You might want to specify first and next extent sizes to reduce the number of tblspace **tblspace** extents and reduce the frequency of situations when you need to place the tblspace **tblspace** extents in non-primary chunks. (A primary chunk is the initial chunk in a dbspace.)

You can use oncheck -pt and oncheck -pT to show the first and next extent sizes of a tblspace **tblspace**.

If you want to configure the first extent for a non-root dbspace, use the onspaces utility.

TBLTBLNEXT configuration parameter

The TBLTBLNEXT configuration parameter specifies the next extent size of tblspace **tblspace** in the root dbspace. Set this parameter if you do not want the database server to automatically manage the extent size.

onconfig.std value

TBLTBLNEXT 0

values

From equivalent of 4 pages specified in kilobytes to the maximum chunk size minus three pages

units

Kilobytes

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

If there is not enough space for a next extent in the primary chunk, the extent is allocated from another chunk. If the specified space is not available, the closest available space is allocated.

TEMPTAB_NOLOG configuration parameter

Use the TEMPTAB_NOLOG configuration parameter to disable logging on temporary tables.

onconfig.std value

TEMPTAB_NOLOG 0

values

`0` = Enable logical logging on temporary table operations

`1` = Disable logical logging on temporary table operations

`2` = Enable logical logging on temporary table operations for primary server and disable logical logging on temporary table operations for secondary servers(HDR, RSS and SDS).

On primary/standard server: same behavior as `0`

On secondary servers: same behavior as `1`

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

This parameter can improve performance in application programs because it prevents HCL OneDB™ from transferring temporary tables over the network. The setting can be updated dynamically with the onmode -wf utility.

If you enable this setting, be aware that because no data is logged when using temporary tables, rolling back a transaction on a temporary table will no longer undo the work in the temporary table.

For HDR, RSS and SDS secondary servers in a high-availability cluster, logical logging on temporary tables should always be disabled by setting the TEMPTAB_NOLOG configuration parameter to 1 or 2.

When server type changes, logging for temporary tables will be enabled/disabled depending on the current server role. It will be effective for temporary tables created afterwards.

TENANT_LIMIT_CONNECTIONS configuration parameter

The TENANT_LIMIT_CONNECTIONS configuration parameter specifies the maximum number of connections to a tenant database.

onconfig.std value

0 (off)

if not present

0 (off)

values

1 - 65536

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

When the limit is reached, subsequent connection requests to the tenant database are rejected.

The `tenant_limit_connections` tenant database property set through the tenant create or tenant update SQL API command takes precedent over the TENANT_LIMIT_CONNECTIONS configuration parameter setting.

TENANT_LIMIT_MEMORY configuration parameter

The TENANT_LIMIT_MEMORY configuration parameter specifies the maximum amount of shared memory for all sessions that are connected to the tenant database.

onconfig.std value

0 (off)

if not present

0 (off)

values

102400 - 2147483648

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

When the limit is exceeded, the session that is using the most shared memory is terminated. The value ranges from 100 MB to 2 TB, but must be specified as an integer that represents the number of KB.

The `tenant_limit_memory` tenant database property set through the tenant create or tenant update SQL administration API command takes precedent over the TENANT_LIMIT_MEMORY configuration parameter setting.

TENANT_LIMIT_SPACE configuration parameter

The TENANT_LIMIT_SPACE configuration parameter specifies the maximum amount of storage space available to a tenant database. Storage space includes all permanent dbspaces, BLOB spaces, and sbspaces.

onconfig.std value

0 (off)

if not present

0 (off)

values

1048576 - 1717986918400

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The TENANT_LIMIT_SPACE configuration parameter limits the amount of permanent storage space available to a tenant database, and can conserve system resources within a tenant-database environment. When the limit is reached, subsequent operations that require additional disk space are rejected. The value ranges from 1 GB to 200 TB, but must be specified as an integer that represents the number of KB.

The `tenant_limit_space` tenant database property set through the tenant create or tenant update SQL administration API command takes precedent over the TENANT_LIMIT_SPACE configuration parameter setting.

TXTIMEOUT configuration parameter

Use the TXTIMEOUT configuration parameter to specify the amount of time that a participant in a two-phase commit waits before it initiates participant recovery. This parameter is used only for distributed queries that involve a remote database server. Nondistributed queries do not use this parameter.

onconfig.std value

TXTIMEOUT 300

values

Positive integers

units

Seconds

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

UNSECURE_ONSTAT configuration parameter

Use the UNSECURE_ONSTAT configuration parameter to remove the database system administrator (DBSA) user access restriction for onstat commands.

onconfig.std value

Not set.

values

`1` = All users can run onstat commands to view running SQL statements

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

By default, the onstat commands that show the SQL statement text from an active session are restricted to DBSA users. To remove this restriction, set the UNSECURE_ONSTAT configuration parameter to `1`. The onstat commands that show SQL statements include onstat -g his, onstat -g ses, onstat -g stm, onstat -g ssc, and onstat -g sql.

UPDATABLE_SECONDARY configuration parameter

Use the UPDATABLE_SECONDARY configuration parameter to set the number of connections to establish between the primary and secondary servers. Setting this configuration parameter enables client applications to perform update, insert, and delete operations on a high-availability secondary server.

onconfig.std value

UPDATABLE_SECONDARY 0

values

Any number from zero (the default value) up to twice the number of CPU VPs. Setting the value to `0` configures the secondary server as read-only. Setting the value from `1` through twice the number of CPU VPs makes the secondary server updatable and configures connection threads.

units

Number of network connections between a given secondary server and its primary server

takes effect

After you edit your `onconfig` file and restart the database server.

Isolation Levels for Secondary Data Replication Servers

If the UPDATABLE_SECONDARY configuration parameter is not set or is set to zero, a secondary data replication server is read-only. In this case, only the DIRTY READ or READ UNCOMMITTED transaction isolation levels are available on secondary servers.

If the UPDATABLE_SECONDARY parameter is set to a valid number of connections greater than zero, a secondary data replication server can support the COMMITTED READ , COMMITTED READ LAST COMMITTED, or COMMITTED READ transaction isolation level, or the USELASTCOMMITTED session environment variable. Only SQL DML statements, such as INSERT, UPDATE, MERGE, and DELETE, and the dbexport utility, can support write operations on an updatable secondary server. (Besides UPDATABLE_SECONDARY, the STOP_APPLY and USELASTCOMMITTED configuration parameters must also be set to enable write operations by dbexport on a secondary data replication server.)

USELASTCOMMITTED configuration parameter

Use the USELASTCOMMITTED configuration parameter to specify the isolation level for which the LAST COMMITTED feature of the COMMITTED READ isolation level is implicitly in effect.

onconfig.std value

USELASTCOMMITTED "NONE"

default value

"NONE"

values

"NONE" = No isolation level identified. If your session encounters an exclusive lock when attempting to read a row in the Committed Read, Dirty Read, Read Committed, or Read Uncommitted isolation level, your transaction cannot read that row until the concurrent transaction that holds the exclusive lock is committed or rolled back.

"COMMITTED READ" = All transactions from a Committed Read isolation level are treated as last committed transactions. The database server reads the most recently committed version of the data when it encounters an exclusive lock while attempting to read a row in the Committed Read or Read Committed isolation level.

"DIRTY READ" = All transactions from a Dirty Read isolation level are treated as last committed transactions. The database server reads the most recently committed version of the data if it encounters an exclusive lock while attempting to read a row in the Dirty Read or Read Uncommitted isolation level.

"ALL" = All transactions from both Committed Read and Dirty Read isolation levels are treated as last committed transactions. database server reads the most recently committed version of the data if it encounters an exclusive lock while attempting to read a row in the Committed Read, Dirty Read, Read Committed, or Read Uncommitted isolation level.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

The LAST COMMITTED feature can reduce the risk of locking conflicts between concurrent transactions on tables that have exclusive row locks. The USELASTCOMMITTED configuration parameter can also enable LAST COMMITTED semantics for READ COMMITTED and READ UNCOMMITTED isolation levels of the SET TRANSACTION statement.

The USELASTCOMMITTED configuration parameter only works with tables that have been created or altered to have ROW as their locking granularity. Tables created without any explicit lock mode setting will use the default setting in DEF_TABLE_LOCKMODE. If DEF_TABLE_LOCKMODE is set to PAGE, the USELASTCOMMITTED configuration parameter cannot enable access to the most recently committed data in tables on which uncommitted transactions hold exclusive locks, unless the tables were explicitly altered to have ROW level of locking granularity.

Use with Shared Disk secondary database servers

The USELASTCOMMITTED configuration parameter is also valid on Shared Disk (SD) secondary database servers. The following table shows valid values for the USELASTCOMMITTED configuration parameter on SD secondary servers and their descriptions.

Table 74. Valid secondary server USELASTCOMMITTED values

USELASTCOMMITTED value	Description
NONE	COMMITTED READ LAST COMMITTED is not the default isolation level for sessions
COMMITTED READ	COMMITTED READ LAST COMMITTED is the default isolation level for all sessions with Committed Read isolation
DIRTY READ	COMMITTED READ LAST COMMITTED is the default isolation level for all sessions with Dirty Read isolation
ALL	COMMITTED READ LAST COMMITTED is the default isolation level for all sessions with Committed Read or Dirty Read isolation

USEOSTIME configuration parameter

Use the USEOTIME configuration parameter to control whether the database server uses subsecond precision when obtaining the current time from the operating system.

onconfig.std value

USEOSTIME 0

values

`0` = Off

`1` = On

takes effect

During initialization

refer to

- Your *HCL OneDB™ Performance Guide*
- Using the CURRENT function to return a datetime value, in the *HCL OneDB™ Guide to SQL: Syntax*.

Usage

Setting USEOSTIME to `1` specifies that the database server is to use subsecond precision when it obtains the current time from the operating system for SQL statements. The following example shows subseconds in a datetime value:

```
2001-09-29 12:50:04.612
```

If subsecond precision is not needed, the database server retrieves the current time from the operating system once per second, making the precision of time for client applications one second. If you set USEOSTIME to `0`, the current function returns a zero (`.000`) for the year to fraction field.

When the host computer for the database server has a clock with subsecond precision, applications that depend on subsecond accuracy for their SQL statements should set USEOSTIME to `1`.

Systems that run with USEOSTIME set to nonzero notice a performance degradation of up to 4 to 5 percent compared to running with USEOSTIME turned off.

This setting does not affect any calls regarding the time from application programs to HCL OneDB™ embedded-language library functions.

USERMAPPING configuration parameter (UNIX™, Linux™)

Use the USERMAPPING configuration parameter to set whether or not the database server accepts connections from mapped users.

default value

`OFF`

values

`OFF` = Only users that are registered in the HCL OneDB™ host computer OS with a login service can connect to the database server. Externally authenticated users without OS accounts on the HCL OneDB™ host computer cannot connect to database server resources.

`BASIC` = Users can connect to HCL OneDB™ without an OS account. A user without an OS account cannot perform privileged user operations on the database server, even if the user maps to a server administrator user or group ID.

`ADMIN` = Users can connect to HCL OneDB™ without an OS account. If a user has authenticated with the identity of a privileged user and is mapped to the proper server administrator group ID, the user can perform DBSA, DBSSO, or AAO work on the database server.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Externally authenticated users without operating system (OS) accounts on the HCL OneDB™ host computer can access database server resources when USERMAPPING is turned on by setting the parameter with the BASIC or ADMIN value. The setting of BASIC or ADMIN also determines whether or not mapped users can be granted administrative privileges.

Important: Changing the USERMAPPING configuration parameter from OFF to ADMIN or BASIC is not the only step in setting up HCL OneDB™ for mapped users. To map users with the appropriate user properties, you must also use DDL statements such as CREATE USER and ALTER USER to register values in appropriate system tables of the SYSUSER database. Depending on the DDL statement used and the defined table mapping, the following tables will be updated or populated:

- SYSINTAUTHUSERS
- SYSUSERMAP
- SYSSURORGATES
- SYSSURROGATEGROUPS

USRC_HASHSIZE configuration parameter

The USRC_HASHSIZE configuration parameter specifies the number of hash buckets in the LBAC credential memory cache. This memory cache holds information about the LBAC credentials of users.

onconfig.std value

USRC_HASHSIZE 31

values

Any positive integer

units

KB

takes effect

After you edit your `onconfig` file and restart the database server.

USRC_POOLSIZE configuration parameter

The USRC_POOLSIZE configuration parameter specifies the maximum number of entries in each hash bucket of the LBAC credential memory cache. This memory cache holds information about the LBAC credentials of users.

onconfig.std value

USRC_POOLSIZE 127

values

A positive value 127 or greater that represents the maximum number of entries in the cache. A positive value 127 or greater that represents half of the initial maximum number of entries in the cache. The maximum value is dependent upon the shared memory configuration and available shared memory for the server instance.

takes effect

After you edit your `onconfig` file and restart the database server.

When you increase the value in memory by running the onmode -wm command.

When you reset the value in memory by running the onmode -wm command.

The initial number of entries in the cache is twice the value of the USRC_POOLSIZE configuration parameter. For example, if the USRC_POOLSIZE configuration parameter is set to 127, 254 entries are allowed in the cache. If all entries in a cache are full, the cache size automatically grows by 10%. To reduce the size of the cache, decrease the value of the USRC_POOLSIZE configuration parameter in the `onconfig` file and restart the server.

USTLOW_SAMPLE configuration parameter

Use the USTLOW_SAMPLE configuration parameter to enable the generation of index statistics based on sampling when you run UPDATE STATISTICS statements in LOW mode.

For an index with more than 100 K leaf pages, the gathering of statistics using sampling can increase the speed of the UPDATE STATISTICS operation.

onconfig.std value

USTLOW_SAMPLE 1

values

`0` = disable sampling

`1` = enable sampling

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

VP_MEMORY_CACHE_KB configuration parameter

Use the VP_MEMORY_CACHE_KB parameter to create a private memory cache for each CPU virtual processor and tenant virtual processor.

onconfig.std value

VP_MEMORY_CACHE_KB 0

values

`0` = Off

The total size of all private memory caches, optionally followed by a comma and the mode of the caches.

Size, in KB:

- 800 to 40% of the SHMTOTAL configuration parameter value.

Mode:

- STATIC = Default. The specified size is the maximum combined size of all private memory caches.
- DYNAMIC = The specified size is the initial size of all private memory caches. The cache size changes dynamically but cannot exceed the value of the SHMTOTAL configuration parameter.

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

Private memory caches can improve performance for memory that is allocated by threads in the HCL OneDB™ server. Private memory caches have no impact on the memory that is allocated to and used by buffer pools or shared memory communication.

When you set the value of the VP_MEMORY_CACHE_KB configuration parameter to a number other than 0, a private memory cache is created for each CPU virtual processor and tenant virtual processor. By default, size of all private memory caches combined is limited to the specified number of KB.

If you want the size of each private memory cache to increase and decrease automatically, as needed, include a comma and the word DYNAMIC after the size, for example, VP_MEMORY_CACHE_KB 1000,DYNAMIC. Although the maximum initial size of all private memory caches combined cannot exceed 40 percent of the value of the SHMTOTAL configuration parameter, with DYNAMIC mode set, the size of the caches can expand beyond the initial limit. The total size of the caches cannot exceed the value of the SHMTOTAL configuration parameter. DYNAMIC mode can improve performance when many threads are disconnecting at the same time or there is contention for the shared memory lock. You can use the onstat -g wmx command to display information about mutexes, such as the shared memory lock mutex `shmcb sh_lock`, and any threads waiting on mutexes.

Attention: Dynamic memory caches on busy systems can grow quickly and use a large amount of available memory. Therefore, if you set the mode to DYNAMIC, set the SHMTOTAL configuration parameter to a specific limit instead of the default value of 0, which does not limit the amount of memory.

If you reset the VP_MEMORY_CACHE_KB configuration parameter to 0, the memory caches are emptied and disabled.

The onstat -g vpcache command returns statistics about private memory caches.

VPCLASS configuration parameter

Use the VPCLASS configuration parameter to create and configure virtual processors.

onconfig.std values

UNIX™: `VPCLASS cpu,num=1,noage`

Windows™:

```
VPCLASS cpu,num=1,noage
#VPCLASS aio,num=1
#VPCLASS jvp,num=1
```

values

Up to 128 bytes of characters. Each VPCLASS configuration parameter value must be unique, begin with a letter or underscore, and contain only digits, letters, underscores, or $ characters. Do not include blank spaces. See the Usage section.

separators

Separate each field with a comma.

takes effect

After you edit your `onconfig` file and restart the database server.

Usage

The VPCLASS configuration parameter has three entries in the `onconfig.std` file, but only the first one is in effect:

- `VPCLASS cpu,num=1,noage`: Creates a CPU virtual processor without aging.
- `#VPCLASS aio,num=1`: Remove the comment symbol to create one AIO virtual processor.
- `#VPCLASS jvp,num=1`: Remove the comment symbol to create one JVP virtual processor.

You can add multiple VPCLASS configuration parameter entries in your `onconfig` file. Each VPCLASS configuration parameter must describe a different class of virtual processors. Put each definition on a separate line.

Syntax for the VPCLASS configuration parameter

VPCLASS { *class* <cpu class> | **aio** **[autotune]** = { **0** | **1** } | *user_defined* [, **noyield**] } [, **num** = *number_vps*] [, **max** = *maximum*]

cpu class

" **cpu** "

" [, **aff** = ({ | *processor* | *start* - *end* [/ *increment*] })] "

" [, **noage**] "

" [, **[autotune]** = { **0** | **1** }] "

Table 75. Options for the VPCLASS configuration parameter value

Field	Values
class	The *class* value is the name of the virtual processor class. The database server starts most virtual processors as needed. Typically, you might set the VPCLASS configuration parameter for the CPU, AIO, JVP, and user-defined virtual processor classes. The virtual processor class name is not case-sensitive. For a list of class names, see Virtual processor classes on page .
user_defined	The *user_defined* value is the name of a virtual processor class that you create for user-defined routines. Make sure the SINGLE_CPU_VP configuration parameter is set to 0.
autotune	Specifies whether the database server adds virtual processors for the specified class as needed to improve performance, up to the value of the max option, if it is included. • autotune=0 prevents the automatic addition of virtual processors • autotune=1 enables the automatic addition of virtual processors If the class is cpu, any CPU virtual processors that are automatically added do not have affinity. The aff option is ignored.
cpu	Specifies the CPU virtual processor class.
num	The *number_vps* value sets the number of virtual processors of the specified class that the database server starts when the database server starts. The default value is 1. The range of

Table 75. Options for the VPCLASS configuration parameter value

(continued)

Field	Values
	values for the cpu and aio virtual processor classes is 1 - 10000. The range of values for all other virtual processor classes is 0 - 10000. You can use the onmode -p command to add virtual processors for the class for the current session.
max	The *maximum* value specifies the maximum number of virtual processors that the database server can start for the class. The value can be any integer greater than 0. By default, the number is unlimited.
aff	On multiprocessor computers that support processor affinity, the aff option specifies the CPUs to which the database server binds CPU virtual processors. The operating system numbers the CPUs from 0 to one less than the number of CPUs. By default, CPU virtual processors are assigned to available processors in round-robin fashion. The aff option takes one or more integers: • *processor* = The CPU number to which to bind the CPU virtual processors. The CPU numbers can be listed in any order. • *start* = The beginning of a range of CPU numbers. • *end* = The end of a range of CPU numbers. • *increment* = A factor that specifies which of the CPU numbers in a range are used. For example, `aff=(1-5/2)` specifies to use CPU numbers 1, 3, and 5.
noage	Disables priority aging for CPU virtual processors, if the operating system implements priority aging. By default, priority aging is in effect.
noyield	Specifies that a user-defined virtual processor class does not yield, which allows the C UDR to yield to other threads that need access to the user-defined virtual processor class. By default, threads for user-defined virtual processors yield. A nonyielding user-defined virtual processors class runs a user-defined routine in a way that gives the routine exclusive use of the virtual processor class. User-defined routines that use a noyield virtual-processor class run serially and never yield the virtual processors to another thread. Specify only one virtual processor in a nonyielding user-defined virtual processor class, because the UDR runs on a single virtual processor until it completes and any additional virtual processors would be idle.

The options can appear in any order, separated by commas.

Use the onmode -p command to dynamically add or remove virtual processors for the current database session. The onmode -p command does not update the `onconfig` file.

CPU virtual processors

On a single-processor computer, allocate only one CPU virtual processor. On a multiprocessor computer, allocate a total number of CPU virtual processes plus user-defined virtual processors up to the number of CPUs on the computer.

When the database server starts, the number of CPU virtual processors is automatically increased to half the number of CPU processors on the database server computer, unless the SINGLE_CPU_VP configuration parameter is enabled.

If you include the autotune option, the database server adds CPU virtual processors as needed to improve performance, up to the number of CPUs on the computer.

The value of the num option of the VPCLASS configuration parameter for the CPU class is not updated when the database server automatically adds CPU virtual processors.

You can configure processor affinity and whether to allow aging. For example, the following entry creates four CPU virtual processors that are bound to CPU numbers 7, 8, 9, and 10, and are not affected by priority aging:

```
VPCLASS CPU,num=4,aff=(7-10),noage
```

AIO virtual processors

Use a VPCLASS configuration parameter entry for the AIO virtual processor class to specify an exact number of AIO virtual processors or to enable the database server to add AIO virtual processors as needed.

Use a VPCLASS configuration parameter entry for the AIO virtual processor class to specify an exact number of AIO virtual processors.

When no VPCLASS configuration parameter entry for the AIO virtual processor class is set, the number of AIO virtual processors is determined by the setting of the AUTO_AIOVPS configuration parameter and is limited to 128:

- If AUTO_AIOVPS is set to 1 (on), the number of AIO virtual processors that are initially started is equal to the number of AIO chunks.
- If AUTO_AIOVPS is set to 0 (off), the number of AIO virtual processors that are started is equal to the greater of 6 or twice the number of AIO chunks.

Java™ virtual processors

If you use Java™ user-defined routines or Java™ applications, create at least one Java™ virtual processor by adding a VPCLASS configuration parameter entry for the JVP virtual processor class. If you set the number of JVPs to zero, or if there is no VPCLASS parameter for the JVP class, you cannot run Java™ UDRs.

VP_KAIO_PERCENT configuration parameter

VP_KAIO_PERCENT is the percentage of total KAIO event resources on the system that each CPU VP will allocate.

onconfig.std value

VP_KAIO_PERCENT 0

if not present

0 (off)

values

1-100

units

Percent

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

Usage

VP_KAIO_PERCENT is the percentage of total KAIO event resources on the system that each CPU VP will allocate.

If you set VP_KAIO_PERCENT to 10 and the total number of KAIO events configured in the OS kernel is 50000 then each CPU VP that starts will attempt to allocate 5000 events.

A value of 0 follows legacy algorithm for allocating KAIO resources to VPs.

WSTATS configuration parameter

Use the WSTATS configuration parameter to specify whether the onstat -g wst command displays wait statistics for threads within the system.

Attention: You should expect a small performance impact due to the cost of gathering statistical information. Enabling the WSTATS configuration parameter for production systems is not recommended.

onconfig.std value

WSTATS 0

range of values

`0` = Disable wait statistics

`1` = Enable wait statistics

takes effect

After you edit your `onconfig` file and restart the database server.

When you reset the value dynamically in your `onconfig` file by running the onmode -wf command.

When you reset the value in memory by running the onmode -wm command.

The sysmaster database

These topics describe the **sysmaster** database and provide reference information for the *system-monitoring interface* (SMI).

These topics include:

- A description of the **sysmaster** database
- Information about how to use SMI tables
- Descriptions of the SMI tables
- A map of the documented SMI tables

For information about the ON-Bar tables, see the *HCL OneDB™ Backup and Restore Guide*.

The sysmaster Database

The database server creates and maintains the **sysmaster** database. It is analogous to the system catalog for databases, which is described in the *HCL OneDB™ Guide to SQL: Reference*. Just as a system catalog for every database managed by the database server keeps track of objects and privileges in the database, a **sysmaster** database for every database server keeps track of information about the database server.

The **sysmaster** database contains the *system-monitoring interface* (SMI) tables. The SMI tables provide information about the state of the database server. You can query these tables to identify processing bottlenecks, determine resource usage, track session or database server activity, and so on. This chapter describes these tables, which are slightly different from ordinary tables.

Warning: The database server relies on information in the **sysmaster** database. Do not change any of the tables in **sysmaster** or any of the data within the tables. Such changes could cause unpredictable and debilitating results.

The database server creates the **sysmaster** database when it initializes disk space. The database server creates the database with unbuffered logging. You cannot drop the database or any of the tables in it, and you cannot turn logging off.

As user **informix** on UNIX™ or a member of the **Informix-Admin** group on Windows™, you can create SPL routines in the **sysmaster** database. (You can also create triggers on tables within **sysmaster**, but the database server never executes those triggers.)

Joins of multiple tables in **sysmaster** might return inconsistent results because the database server does not lock the tables during a join. You can join **sysmaster** tables with tables in other databases. However, to join **sysmaster** tables with tables in a nonlogging database, first make the nonlogging database the current database.

The buildsmi Script

When you bring the database server up for the first time, it runs a script called **buildsmi**, which is in the **etc** directory. This script builds the database and tables that support SMI. The database server requires approximately 1750 free pages of logical-log space to build the **sysmaster** database.

If you receive an error message that directs you to run the **buildsmi** script, a problem probably occurred while the database server was building the SMI database, tables, and views. When you use **buildsmi**, the existing **sysmaster** database is dropped and then re-created.

This script must be run as user **informix** on UNIX™, or as a member of the **Informix-Admin** group on Windows™, after ensuring that no connections to the sysmaster database are made during the build of the database. For example, if a Scheduler task is running when the **buildsmi** script commences, the script fails when the Scheduler attempts to access any of the sysmaster tables.

Errors issued while the **buildsmi** script runs are written (on UNIX™) to the file **/tmp/buildsmi.out**, or on Windows™ to the file **%ONEDB_HOME%\etc\buildsmi_out.%*ONEDB_SERVER*%**, where ***%ONEDB_SERVER%*** is the name of the HCL OneDB™ instance.

The bldutil.sh Script

When you initialize the database server for the first time, it runs a script called **bldutil.sh** on UNIX™ or **bldutil.bat** on Windows™. This script builds the **sysutils** database. If it fails, the database server creates an output file in the **tmp** directory. The output file is **bldutil.process_id** on UNIX™ and **bldutil.out** on Windows™. The messages in this output file reflect errors that occurred during the script execution.

The System-Monitoring Interface

This section describes the SMI tables and how you access them to monitor the database server operation.

Understanding the SMI Tables

The SMI (system-monitoring interface) consists of tables and pseudo-tables that the database server maintains automatically. While the SMI tables appear to the user as tables, they are not recorded on disk as normal tables are. Instead, the database server constructs the tables in memory, on demand, based on information in shared memory at that instant. When you query an SMI table, the database server reads information from these shared-memory structures. Because the database server continually updates the data in shared memory, the information that SMI provides lets you examine the current state of your database server.

The SMI tables provide information about the following topics:

- Auditing
- Checkpoints
- Chunk I/O
- Chunks
- Database-logging status

- Dbspaces
- Disk usage
- Environment variables
- Extents
- Locks
- Networks
- SQL statement cache statistics
- SQL statements
- System profiling
- Tables
- User profiling
- Virtual-processor CPU usage

The data in the SMI tables changes dynamically as users access and modify databases that the database server manages.

Accessing SMI tables

Any user can use SQL SELECT statements to query an SMI table, but standard users cannot run statements other than the SELECT statement. Users who attempt to run other statements result in permission errors. The administrator can run SQL statements other than SELECT, but the results of such statements are unpredictable.

Tip: For more predictable results, query the views that are associated with each table instead of querying the tables directly.

If you query the **systabpaghdrs** table directly, you must specify an appropriate value for the `pg_partnum` parameter. The value is `pg_partnum > 1048576`. However, if you query the view that is associated with the **systabpaghdrs** table, you do not have to specify this value for the `pg_partnum` parameter.

HCL OneDB™ includes the **sysadtinfo** and **sysaudit** tables. Only the user **informix** on UNIX™ or members of the **Informix-Admin** group on Windows™ can query the **sysadtinfo** and **sysaudit** tables.

You cannot use the **dbschema** or **dbexport** utilities on any of the tables in the **sysmaster** database. If you do, the database server generates the following error message:

```
Database has pseudo tables - can't build schema
```

SELECT statements

You can use SELECT statements on SMI tables wherever you can use SELECT against ordinary tables.

For example, you can use SELECT statements ordinary tables from DB-Access, in an SPL routine, with , and so on.

Restriction: You cannot meaningfully reference **rowid** when you query SMI tables. SELECT statements that use **rowid** do not return an error, but the results are unpredictable.

All standard SQL syntax, including joins between tables, sorting of output, and so on, works with SMI tables. For example, if you want to join an SMI table with a non-SMI table, name the SMI table with the following standard syntax:

```
sysmaster[@dbservername]:[owner.]tablename
```

Triggers and Event Alarms

Triggers based on changes to SMI tables never run. Although you can define triggers on SMI tables, triggers are activated only when an INSERT, UPDATE, or DELETE statement occurs on a table. The updates to the SMI data occur within the database server, without the use of SQL, so a trigger on an SMI table is never activated, even though the data returned by a SELECT statement indicates that it should be.

To create an event alarm, query for a particular condition at predefined intervals, and execute an SPL routine if the necessary conditions for the alarm are met.

SPL and SMI Tables

You can access SMI tables from within a SPL routine. When you reference SMI tables, use the same syntax that you use to reference a standard table.

Locking and SMI Tables

The information in the SMI tables changes based on the database server activity. However, the database server does not update the information using SQL statements. When you use SMI tables with an isolation level that locks objects, it prevents other users from accessing the object, but it does not prevent the data from changing. In this sense, all the SMI tables have a permanent Dirty Read isolation level.

The System-Monitoring Interface Tables

The **sysmaster** database contains many tables that you can use to monitor your system.

Tip: For each system-monitoring interface (SMI) table, there is a corresponding view with the same name. For the best results, query the views that are associated with tables instead of querying the underlying tables directly.

Many other tables in the **sysmaster** database are part of the system-monitoring interface but are not documented. Their schemas and column content can change from version to version. The **flags_text** table now contains more rows. To view the new rows, first drop and then re-create the **sysmaster** database.

The following table lists the SMI tables.

Table 76. SMI tables

Table	Description	Reference
sysadtinfo	Auditing configuration information	sysadtinfo on page 225
sysaudit	Auditing event masks	sysadtinfo on page 225
syscheckpoint	Checkpoint information	syscheckpoint on page 226
syschkio	Chunk I/O statistics	syschkio on page 226
syschunks	Chunk information	syschunks on page 227
syscluster	High-availability cluster information	syscluster on page 231
syscmsmsla	Connection Manager information	syscmsmsla on page 233
syscmsmtab	Connection Manager information	syscmsmtab on page 233
syscmsmunit	Information for each Connection Manager unit in a Connection Manager configuration file	syscmsmunit on page 234
syscompdicts_full	Compression dictionary information	syscompdicts_full on page 234
sysconfig	Configuration information	sysconfig on page 236
sysdatabases	Database information	sysdatabases on page 236
sysdbslocale	Locale information	sysdbslocale on page 237
sysdbspaces	Dbspace information	sysextents on page 240
sysdri	Data-replication information	sysdri on page 239
sysdual	Is a single-row table	sysdual on page 240
sysenv	Server startup environment	sysenv on page 240

Table 76. SMI tables (continued)

Table	Description	Reference
sysenvses	Session-level environment variable	sysenvses on page 240
sysextents	Extent-allocation information	sysextents on page 240
sysextspaces	External spaces information	sysextspaces on page 241
sysha_lagtime	Secondary server lag-time statistics	sysha_lagtime Table on page 242
sysha_type	Information about connected servers	sysha_type on page 244
sysha_workload	Secondary server workload statistics	sysha_workload on page 245
sysipl	Index page logging information	sysipl on page 246
syslocks	Active locks information	syslocks on page 246
syslogs	Logical-log file information	syslogs on page 247
syslogfil	System log file information	syslogfil table on page 248
sysmgminfo	Memory Grant Manager and Parallel Data Query information	sysmgminfo on page 249
sysnetclienttype	Client type network activity	sysnetclienttype on page 250
sysnetglobal	Global network information	sysnetglobal on page 251
sysnetworkio	Network I/O	sysnetworkio table on page 251
sysonlinelog	Online log information	sysonlinelog on page 252
sysprofile	System-profile information	sysprofile on page 252
sysproxyagents	Information about all the proxy agent threads	sysproxyagents on page 254

Table 76. SMI tables (continued)

Table	Description	Reference
sysproxydistributors	Proxy distributor information	sysproxydistributors on page 255
sysproxysessions	Information about sessions that use updatable secondary servers	sysproxysessions table on page 255
sysproxytxnops	Information about transactions that are run through each proxy distributor	sysproxytxnops table on page 256
sysproxytxns	Information about all of the current transactions that run through each proxy distributor	sysproxytxns table on page 257
sysptprof	Table information	sysptprof table on page 259
sysrsslog	RS secondary server information	sysrsslog on page 265
sysscblst	Memory by user	sysscblst on page 265
syssesprof	Counts of various user actions	syssesprof on page 266
syssesappinfo	Distributed Relational Database Architecture™ (DRDA®) client-session information.	syssesappinfo on page 266
syssessions	Description of each user connected	syssessions on page 267
syssessiontempspace usage	Contains information about each session's temp space usage.	syssessiontempspace usage on page 269
syssmx	SMX (server multiplexer group) connection information	syssmx on page 270
syssmxses	SMX (server multiplexer group) session information	syssmxses on page 270
syssqexplain	SQL statement information that is enabled by the SET EXPLAIN statement	syssqexplain table on page 270
syssqltrace	SQL statement information	syssqltrace on page 272
syssqltrace_hvar	SQL statement tracing host variable information	syssqltrace_hvar on page 274
syssqltrace_info	SQL profile trace system information	syssqltrace_info on page 274

Table 76. SMI tables (continued)

Table	Description	Reference
syssqltrace_iter	SQL statement iterators	syssqltrace_iter on page 275
syssrcrss	RS secondary server statistics	syssrcrss on page 275
syssrcsds	SD secondary server statistics	syssrcsds on page 276
systabnames	Database, owner, and table name for the tblspace **tblspace**	systabnames on page 277
systabpaghdrs	Page headers	None
systhreads	Wait statistics	systhreads on page 277
systrgrss	RS secondary server statistics	systrgrss on page 278
systrgsds	SD secondary server statistics	systrgsds on page 278
sysvpprof	User and system CPU used by each virtual processor	sysvpprof on page 279

The sysutils Tables

ON-Bar uses the following tables in the **sysutils** database. For more information, see the *HCL OneDB™ Backup and Restore Guide*.

Table

Description

bar_action

Lists all backup and restore actions that are attempted against an object, except during a cold restore. Use the information in this table to track backup and restore history.

bar_instance

Writes a record to this table for each successful backup. ON-Bar might later use the information for a restore operation.

bar_object

Describes each backup object. This table provides a list of all storage spaces and logical logs from each database server for which at least one backup attempt was made.

bar_server

Lists the database servers in an installation. This table is used to ensure that backup objects are returned to their proper places during a restore.

sysadtinfo

The **sysadtinfo** table contains information about the auditing configuration for the database server. For more information, see your *HCL OneDB™ Security Guide*. You must be user **informix** or user **root** on UNIX™ or a member of the **Informix-Admin** group on Windows™ to retrieve information from the sysadtinfo table.

Column	Type	Description
adtmode	integer	Controls the level of auditing.
adterr	integer	Specifies how the database server behaves when it encounters an error while it writes an audit record.
adtsize	integer	Maximum size of an audit file
adtpath	char(256)	Directory where audit files are written
adtfile	integer	Number of the audit file

sysaudit

For each defined audit mask (that is, for each *username*), the **sysaudit** table contains flags that represent the database events that generate audit records. The **success** and **failure** columns represent the bitmasks that compose the audit masks. If a bit is set in both the **success** the and **failure** columns, the corresponding event generates an audit record whether or not the event succeeded.

You must be user **informix** or **user** root on UNIX™ or a member of the **Informix-Admin** group on Windows™ to retrieve information from the **sysaudit** table.

Use the **onaudit** utility to list or modify an audit mask. For information about **onaudit** and auditing, see your *HCL OneDB™ Security Guide*.

Column	Type	Description
username	char(32)	Name of the mask
succ1	integer	Bitmask of the audit mask for success
succ2	integer	Bitmask of the audit mask for success
succ3	integer	Bitmask of the audit mask for success
succ4	integer	Bitmask of the audit mask for success
succ5	integer	Bitmask of the audit mask for success

Column	Type	Description
fail1	integer	Bitmask of the audit mask for failure
fail2	integer	Bitmask of the audit mask for failure
fail3	integer	Bitmask of the audit mask for failure
fail4	integer	Bitmask of the audit mask for failure
fail5	integer	Bitmask of the audit mask for failure

syschkio

The **syschkio** system-monitoring interface table provides I/O statistics for individual chunks that the database server manages.

Column	Type	Description
chunknum	smallint	Chunk number
reads	integer	Number of physical reads
pagesread	integer	Number of pages read
writes	integer	Number of physical writes
pageswritten	integer	Number of pages written
mreads	integer	Number of physical reads (mirror)
mpagesread	integer	Number of pages read (mirror)
mwrites	integer	Number of physical writes (mirror)
mpageswritten	integer	Number of pages written (mirror)

syscheckpoint

The **syscheckpoint** table provides information and statistics about checkpoints.

Column	Type	Description
interval	integer	Number of checkpoints since the server was started
type	char(12)	Hard or Interval
caller	char(10)	Caller of the checkpoint
clock_time	integer	Time of day the checkpoint occurred
crit_time	float	Time spent waiting for the critical section to be released
flush_time	float	Time spent flushing pages to disk

Column	Type	Description
cp_time	float	Duration from checkpoint pending until checkpoint done
n_dirty_buffs	integer	Number of dirty buffers
plogs_per_sec	integer	Number of physical log pages processed in a second
llogs_per_sec	integer	Number of logical log pages processed in a second
dskflush_per_sec	integer	Number of buffer pool pages flushed in a second
ckpt_logid	integer	Unique id of the logical log at the checkpoint
ckpt_logpos	integer	Position of the logical log at the checkpoint
physused	integer	Number of pages used in the physical log
logused	integer	Number of pages used in the logical log
n_crit_waits	integer	Number of users who had to wait to enter a critical section
tot_crit_wait	float	Duration spent waiting for all users waiting at the checkpoint critical section block
longest_crit_wait	float	Longest critical section wait
block_time	float	Duration of the checkpoint that blocked the system

syschunks

The **syschunks** table contains a description of each of the chunks that the database server manages.

In the **flags** and **mflags** columns, each bit position represents a separate flag. Thus, it might be easier to read values in the **flags** and **mflags** columns if the values are returned by the HEX function.

Table 77. The syschunks table

Column	Type	Description
chknum	smallint	Chunk number
dbsnum	smallint	Dbspace number
nxchknum	smallint	Number of the next chunk in this dbspace
chksize	integer	Number of pages in this chunk (in units of system default page size)
offset	integer	Page offset of the chunk in its device or path
pagesize	integer	Page size (in bytes)
nfree	integer	Number of free pages in the chunk

Table 77. The syschunks table

(continued)

Column	Type	Description
		The amount of free space depends on the type of space: • dbspace = multiply the number of free pages by the system default page size of either 2 KB or 4 KB. • blobspace = multiply the number of free pages by the blobpage size. • sbspace = multiply the number of free pages by the sbpage size (which is the same as the system default page size).
is_offline	integer	1 = chunk is offline 0 = chunk is online
is_recovering	integer	1 = the chunk is being recovered 0 = the chunk is not being recovered
is_blobchunk	integer	1 = the chunk is in a blobspace 0 = the chunk is not in a blobspace
is_sbchunk	integer	1 = the chunk is an sbspace 0 = the chunk is not in an sbspace
is_inconsistent	integer	1 = the chunk is undergoing logical restore 0 = the chunk is not being restored
is_extendable	integer	1 = the chunk is extendable 0 = the chunk is not extendable
flags	smallint	The flags have the following numeric and hexadecimal values and meanings: • 16 (0x0010) = Chunk is a mirrored chunk • 32 (0x0020) = Chunk is in offline mode • 64 (0x0040) = Chunk is in online mode • 128 (0x0080) = Chunk is in recovery mode • 256 (0x0100) = Chunk is mirrored

Table 77. The syschunks table

(continued)

Column	Type	Description
		• 512 (0x0200) = Chunk is part of a blobspace • 1024 (0x0400) = Chunk is being dropped • 2048 (0x0800) = Chunk is part of an optical stageblob • 4096 (0x1000) = Chunk is inconsistent • 8192 (0x2000) = Chunk is extendable • 16384 (0x4000) = Chunk was added during roll forward • 32768 (0x8000) = Chunk was renamed • 65536 (0x10000) = Chunk uses big chunk page header • 131072 (0x20000) = Chunk has a tblspace tblspace extent • 262144 (0x40000) = No checkpoint was completed since this chunk was initialized (primarily for internal use)
fname	char(256)	Path name for the file or device of this chunk
mdsize	integer	Size in pages of the metadata area of a chunk that belongs to an sbspace. -1 = the chunk does not belong to an sbspace.
mfname	char(256)	Path name for the file or device of the mirrored chunk, if any
moffset	integer	Page offset of the mirrored chunk
mis_offline	integer	1 = mirror is offline 0 = mirror is online
mis_recovering	integer	1 = mirror is being recovered 0 = mirror is not being recovered
mflags	smallint	Mirrored chunk flags; values and meanings are the same as the **flags** column.
udfree	integer	Free space in pages within the user data area of a chunk that belongs to an sbspace.

Table 77. The syschunks table

(continued)

Column	Type	Description
		-1 = the chunk does not belong to an sbspace.
udsize	integer	Size in pages of the user data area of a chunk that belongs to an sbspace. -1 = the chunk does not belong to an sbspace.

sysckptinfo

The **sysckptinfo** system-monitoring interface table provides historical information about the previous twenty checkpoints.

Column	Type	Description
ckpt_status	int	0x0011 = A checkpoint was blocked because the physical log ran out of resources. 0x0021 = A checkpoint was blocked because the logical log ran out of resources. 0x0041 = A checkpoint was blocked because transactions were too long. 0x1000 = The physical log is too small. 0x2000 = The logical log space is too small. 0x4000 = The physical log is too small for RTO.
plogs_per_S	int	Average rate of physical logging activity.
llogs_per_S	int	Average rate of logical logging activity.
dskF_per_S	int	Average rate of pages flushed to disk.
longest_dskF	int	Longest duration of time to flush the buffer pool to the disk during checkpoint processing.
dirty_pgs_S	int	Average rate of pages being modified.
sug_plog_sz	int	Suggested physical log size.
sug_llog_sz	int	Suggested logical log space size.
ras_plog_sp	int	Rate at which fast recovery can restore the physical log.
ras_llog_sp	int	Rate at which fast recovery can replay the logical log.

Column	Type	Description
boottime	int	Time it takes for the server to boot shared memory and open chunks.
auto-ckpts	int	`1` = on, `0` = off.
auto_lru	int	`1` = on, `0` = off.
cur_intvl	int	Current® checkpoint interval id.
auto_aiovp	int	`1` = on, `0` = off.

syscluster

The **syscluster** system catalog table stores information about servers in a high-availability cluster. The **syscluster** table has the following columns.

Table 78. syscluster table information

Column	Type	Explanation
name	CHAR(128)	The name of the primary server.
role	CHAR(1)	Code to indicate whether the server is a primary server or secondary server.
syncmode	CHAR(8)	The synchronization mode between the primary server and the secondary server: sync or async.
nodetype	CHAR(8)	The type of server: HDR, RSS, or SDS.
supports_updates	CHAR(1)	Indicates whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the UPDATABLE_SECONDARY configuration parameter).
server_status	CHAR(32)	Indicates the status of the secondary server.
connection_status	CHAR(32)	Indicates the connection status of the secondary server.
delayed_apply	INTEGER	Indicates whether the secondary server waits for a specified amount of time before applying logs (as specified by the DELAY_APPLY configuration parameter).
stop_apply	CHAR(24)	Indicates whether the secondary server has stopped applying log files received from the primary server (as specified by the STOP_APPLY configuration parameter).
logid_sent	INTEGER	Indicates the log ID of the most recent log page sent by the primary server to the secondary server.

Table 78. syscluster table information

(continued)

Column	Type	Explanation
logpage_sent	INTEGER	Indicates the page number of the most recent log page sent by the primary server to the secondary server.
logid_acked	INTEGER	Indicates the log ID of the most recent log page the secondary server acknowledged.
logpage_acked	INTEGER	Indicates the page number of the most recent log page the secondary server acknowledged.
ack_time	DATETIME YEAR TO SECOND	Indicates the date and time of the last acknowledged log.
sdscycle	INTEGER	Indicates the cycle number to which the primary server has advanced. Used internally by HCL support to monitor coordination of the primary server with the secondary server.
sdscycle_acked	INTEGER	Indicates the cycle number that the shared disk secondary server has acknowledged. Used internally by HCL support to monitor coordination of the primary server with the secondary server.

syscmsm

The **syscmsm** table is a view of the **syscmsmtab** and **syscmsmsla** tables. It contains Connection Manager service level agreement (SLA) information. The table is updated one time every five seconds.

Table 79. syscmsm table information

Column	Type	Description
sid	integer	Connection Manager session ID
name	char(128)	Connection Manager name
host	char(256)	Host name
unit	char(128)	Unit name
type	char(128)	Unit type
servers	char(1024)	Unit servers

Table 79. syscmsm table information

(continued)

Column	Type	Description
foc	char(128)	Failover configuration (FOC)
flag	integer	Arbitrator flag. A value of 1 indicates that the Connection Manager Arbitrator is active. A value of 0 indicates that the Arbitrator is inactive.
sla_name	char(128)	SLA name
sla_define	char(128)	SLA definition
connections	integer	Number of connections that are made through Connection Manager

syscmsmsla

The **syscmsmsla** table contains Connection Manager service level agreement (SLA) information. The table is updated one time every five seconds.

Table 80. syscmsmsla table information

Column	Type	Description
address	int8	CMSLA internal address
sid	integer	Connection Manager session ID
sla_name	char(128)	SLA name
sla_define	char(128)	SLA define
connections	integer	Number of connections made through Connection Manager

syscmsmtab

The **syscmsmtab** table contains Connection Manager information.

Table 81. syscmsmtab table information

Column	Type	Description
address	int8	Connection Manager internal address

Table 81. syscmsmtab table information

(continued)

Column	Type	Description
sid	integer	Connection Manager session ID
name	char(128)	Connection Manager name
host	char(256)	Host name
flag	integer	Arbitrator flag. A value of 1 indicates that the Connection Manager Arbitrator is active. A value of 0 indicates that the Arbitrator is inactive.

syscmsmunit

The **syscmsmunit** table contains information for each Connection Manager unit in a Connection Manager configuration file.

Table 82. syscmsmunit table information

Column	Type	Description
address	int8	Connection Manager internal address
sid	integer	Connection Manager session ID
unit	char(128)	Unit name
type	char(128)	Unit type
servers	char(1024)	Unit servers
foc	char(128)	Failover configuration (FOC)
flag	integer	Arbitrator flag. A value of 1 indicates that the Connection Manager Arbitrator is active. A value of 0 indicates that the Arbitrator is inactive.

syscompdicts_full

The **syscompdicts_full** table and the **syscompdicts** view provide information on all compression dictionaries. The only difference between the table and the view is that, for security purposes, the view does not contain the **dict_dictionary** column.

Only user **informix** can retrieve information from the **syscompdicts_full** table. The **syscompdicts** view is not restricted to user **informix**.

The following table shows the information that the **syscompdicts_full** table and the **syscompdicts** view provide for each compression dictionary.

Table 83. Compression Dictionary Information

Column	Type	Description
dict_partnum	integer	Partition number to which the compression dictionary applies
dict_code_version	integer	Version of the code that is creating the compression dictionary `1` is the first version.
dict_dbsnum	integer	Number of the dbspace that the dictionary resides in
dict_create_timestamp	integer	Timestamp that shows when the dictionary was created
dict_create_loguniqid	integer	Unique ID for the logical log that was active when the dictionary was created
dict_create_logpos	integer	Position within the logical log when the dictionary was created
dict_drop_timestamp	integer	Timestamp that shows when the dictionary was dropped.
dict_drop_loguniqid	integer	Unique ID for the logical log that was created when the dictionary was dropped.
dict_drop_logpos	integer	Position within the logical log when the dictionary was dropped.
dict_dictionary	byte	Compression dictionary binary object This column is not included in the **syscompdicts** view.

Example

Sample syscompdicts information

A row of information in the **syscompdicts** view could displays columns containing this information:

```
dict_partnum        1048939
dict_code_version   1
dict_dbsnum         1
dict_create_times+  1231357656
```

```
dict_create_logun+  11
dict_create_logpos  1695768
dict_drop_timesta+  0
dict_drop_loguniq+  0
dict_drop_logpos    0
```

You can use an UNLOAD statement to unload the compression dictionary to a compression dictionary file, as follows:

```
UNLOAD TO 'compression_dictionary_file'
   SELECT * FROM sysmaster:syscompdicts_full;
```

sysconfig

The **sysconfig** table describes the effective, original, and default values of the configuration parameters. For more information about the ONCONFIG file and the configuration parameters, see Database configuration parameters on page 3.

Column	Type	Description
cf_id	integer	Unique numeric identifier
cf_name	char(128)	Configuration parameter name
cf_flags	integer	Reserved for future use
cf_original	char(256)	Value in the ONCONFIG file at boot time
cf_effective	char(256)	Value currently in use
cf_default	char(256)	Value provided by the database server if no value is specified in the ONCONFIG file

sysdatabases

The **sysdatabases** view describes each database that the database server manages.

Table 84. sysdatabases view information

Column	Type	Description
name	char(128)	Database name
partnum	integer	The partition number (tblspace identifier) for the systables table for the database
owner	char(32)	User ID of the creator of the database
created	date	Date created
is_logging	integer	1 If logging is active, 0 if not

Table 84. sysdatabases view information

(continued)

Column	Type	Description	
is_buff_log	integer	1 If buffered logging, 0 if not	
is_ansi	integer	1 If ANSI/ISO-compliant, 0 if not	
is_nls	integer	1 If GLS-enabled, 0 if not	
is_case_insens	integer	1 If case-insensitive for NCHAR and NVARCHAR columns, 0 if not	
flags	smallint	Logging flags (hex values)	
		0	No logging
		1	Unbuffered logging
		2	Buffered logging
		4	ANSI/ISO-compliant database
		8	Read-only database
		10	GLS database
		20	Checking of the logging mode of **syscdr** database bypassed
		100	Changed status to buffered logging
		200	Changed status to unbuffered logging
		400	Changed status to ANSI/ISO compliant
		800	Database logging turned off
		1000	Long ID support enabled

sysdbslocale

The **sysdbslocale** table lists the locale of each database that the database server manages.

Table 85. sysdbslocale table information

Column	Type	Description
dbs_dbsname	char(128)	Database name
dbs_collate	char(32)	The locale of the database

sysdbspaces

The **sysdbspaces** table contains a description of each of the storage spaces that the database server manages.

Table 86. sysdbspaces table information

Column	Type	Description
dbsnum	smallint	Space number
name	char(128)	Space name
owner	char(32)	User ID of owner of the space
fchunk	smallint	Number of the first chunk in the space
nchunks	smallint	Number of chunks in the space
create_size	decimal	The minimum size of a chunk that can be created for this space using the storage pool.
extend_size	decimal	The minimum size by which a chunk in this storage space can be extended, either manually or automatically.
pagesize	integer	Page size
is_mirrored	integer	1 = The space is mirrored 0 = The space is not mirrored
is_blobspace	integer	1 = The space is a blobspace 0 = The space is not a blobspace
is_sbspace	integer	1 = The space is an sbspace 0 = The space is not an sbspace
is_temp	integer	1 = The space is a temporary dbspace 0 = The space is not a temporary dbspace
is_encrypted	integer	1 = The space is encrypted 0 = The space is not encrypted
flags	smallint	Each bit position represents a separate flag. Thus, it might be easier to read values in this column if you return the values with the HEX function.

Table 86. sysdbspaces table information

(continued)

Column	Type	Description
		• 1 (0x0001) = The space has no mirror • 2 (0x0002) = The space uses mirroring • 4 (0x0004) = Mirroring is disabled • 8 (0x0008) = The space is newly mirrored • 16 (0x0010) = The space is a blobspace • 32 (0x0020) = The blobspace is on removable media • 128 (0x0080) = The blobspace has been dropped • 512 (0x0200) = The space is being recovered • 1024 (0x0400) = The space has been physically recovered • 2048 (0x0800) = The space is in logical recovery • 32768 (0x8000) = The space is an sbspace

sysdri

The **sysdri** table provides information about the High-Availability Data-Replication status of the database server.

Column	Type	Description
type	char(50)	High-Availability Data Replication type Possible values: • primary • secondary • standard • not initialized
state	char(50)	State of High-Availability Data Replication Possible values: • off • on • connecting • failure • read-only
name	char(128)	The name of the other database server in the High-Availability Data-Replication pair

Column	Type	Description
intvl	integer	The High-Availability Data-Replication interval
timeout	integer	The High-Availability Data-Replication timeout value for this database server
lostfound	char(256)	The pathname to the lost-and-found file

sysdual

The **sysdual** table returns exactly one column and one row.

Column	Type	Description
dummy	char(1)	Dummy columns returning "X"

sysenv

The **sysenv** table displays the startup environment settings of the database server.

Column	Type	Description
env_id	integer	Identifier variable number
env_name	char(128)	Environment variable name
env_value	char(512)	Environment variable value

sysenvses

The **sysenvses** table displays the environment variable at the session level.

Column	Type	Description
envses_sid	integer	Session id
envses_id	integer	Identifier variable number
envses_name	char(128)	Session environment variable name
envses_value	char(512)	Session environment variable value

sysextents

The **sysextents** table provides information about extent allocation.

Column	Type	Description
dbsname	char(128)	Database name
tabname	char(128)	Table name

Column	Type	Description
chunk	integer	Chunk number
offset	integer	Number of pages into the chunk where the extent begins
size	integer	Size of the extent, in pages

sysextspaces

The sysextspaces table provides information about external spaces. Indexes for the **id** column and the **name** column allow only unique values.

Column	Type	Description
id	integer	External space ID
name	char(128)	External space name
owner	char(32)	External space owner
flags	integer	External space flags (reserved for future use)
refcnt	integer	External space reference count.
locsize	integer	Size of external space location, in bytes
location	char (256)	Location of external space

sysfeatures

The **sysfeatures** view provides general information about various features of the HCL OneDB™ database server instance. The **sysfeatures** view is created from an internal table named **syslicenseinfo**, which is stored permanently on the disk. When the database server instances are initialized, the table is pre-allocated with a fixed size which allows tracking of 260 weeks of data. The data wraps every five years.

Metrics are sampled every 15 minutes and only the highest values during the particular week are stored. Each row in the table contains data only for the specific week it represents.

Column	Type	Description
week	smallint	The week that the information was recorded.
year	smallint	The year that the information was recorded.
version	char(12)	The HCL OneDB™ server version.
max_cpu_vps	smallint	The maximum number of CPU virtual processors.
max_vps	smallint	The maximum number of virtual processors.

Column	Type	Description
max_conns	integer	The maximum number of concurrent physical connections on a standalone or high-availability cluster primary server instance.
max_sec_conns	integer	The maximum number of concurrent physical connections on an HDR secondary or RS secondary server instance.
max_sds_clones	smallint	The maximum number of SD secondary server instances connected to the primary server.
max_rss_clones	smallint	The maximum number of RS secondary server instances connected to the primary server.
total_size	integer	The maximum disk space allocated in all chunks (in megabytes).
total_size_used	integer	The maximum disk space used in all chunks (in megabytes).
max_memory	integer	The maximum memory allocated in all segments (in megabytes).
max_memory _used	integer	The maximum memory used in all segments (in megabytes).
is_primary	integer	Indicates whether the server was a primary server in a particular week; 1 = yes, 0 = no.
is_secondary	integer	Indicates whether the server was an HDR secondary server in a particular week; 1 = yes, 0 = no.
is_sds	integer	Indicates whether the server was an SD secondary server in a particular week; 1 = yes, 0 = no (not implemented; always 0).
is_rss	integer	Indicates whether the server was an RS secondary server in a particular week; 1 = yes, 0 = no.
is_er	integer	Indicates whether the server was an enterprise replication server in a particular week; 1 = yes, 0 = no.
is_pdq	integer	Indicates whether the PDQ feature was used on the server instance in the particular week; 1 = yes, 0 = no.

sysha_lagtime Table

The **sysha_lagtime** table provides a history of the amount of time that it took to apply a log record on any of the secondary nodes.

The **sysha_lagtime** table contains a history of the last 20 samplings performed for a particular secondary server.

Table 87. sysha_lagtime table information

Column	Type	Description
lt_secondary	CHAR(128)	Name of secondary server
lt_time_last_update	INTEGER	Time at which log record was last updated
lt_lagtime_1	FLOAT	Amount of time required to apply log record for the most recent five-second interval
lt_lagtime_2	FLOAT	Amount of time required to apply log record for the second most recent five-second interval
lt_lagtime_3	FLOAT	Amount of time required to apply log record for the third most recent five-second interval
lt_lagtime_4	FLOAT	Amount of time required to apply log record for the fourth most recent five-second interval
lt_lagtime_5	FLOAT	Amount of time required to apply log record for the fifth most recent five-second interval
lt_lagtime_6	FLOAT	Amount of time required to apply log record for the sixth most recent five-second interval
lt_lagtime_7	FLOAT	Amount of time required to apply log record for the seventh most recent five-second interval
lt_lagtime_8	FLOAT	Amount of time required to apply log record for the eighth most recent five-second interval
lt_lagtime_9	FLOAT	Amount of time required to apply log record for the ninth most recent five-second interval
lt_lagtime_10	FLOAT	Amount of time required to apply log record for the tenth most recent five-second interval
lt_lagtime_11	FLOAT	Amount of time required to apply log record for the eleventh most recent five-second interval
lt_lagtime_12	FLOAT	Amount of time required to apply log record for the twelfth most recent five-second interval
lt_lagtime_13	FLOAT	Amount of time required to apply log record for the thirteenth most recent five-second interval
lt_lagtime_14	FLOAT	Amount of time required to apply log record for the fourteenth most recent five-second interval

Table 87. sysha_lagtime table information

(continued)

Column	Type	Description
lt_lagtime_15	FLOAT	Amount of time required to apply log record for the fifteenth most recent five-second interval
lt_lagtime_16	FLOAT	Amount of time required to apply log record for the sixteenth most recent five-second interval
lt_lagtime_17	FLOAT	Amount of time required to apply log record for the seventeenth most recent five-second interval
lt_lagtime_18	FLOAT	Amount of time required to apply log record for the eighteenth most recent five-second interval
lt_lagtime_19	FLOAT	Amount of time required to apply log record for the nineteenth most recent five-second interval
lt_lagtime_20	FLOAT	Amount of time required to apply log record for the twentieth most recent five-second interval

sysha_type

The **sysha_type** table is a single row table that is used to describe the type of server that is connected.

Table 88. sysha_type table information

Column	Type	Description
ha_type	integer	Server type (see table below)
ha_primary	char(128)	Server name (see table below)

Table 89. Descriptions for the values in the sysha_type table

Value of *ha_type*	Value of *ha_primary*	Description
0	NULL	Not part of a high-availability environment
1	*<primary server name>*	Primary server
2	*<primary server name>*	HDR secondary server

Table 89. Descriptions for the values in the sysha_type table

(continued)

Value of *ha_type*	Value of *ha_primary*	Description
3	*<primary server name>*	SD secondary server
4	*<primary server name>*	RS secondary server

sysha_workload

The **sysha_workload** table contains workload statistics on each of the secondary servers.

Table 90. sysha_workload table information

Column	Type	Description
wl_secondary	char(128)	Name of secondary server
wl_time_last_update	integer	Time at which workload last updated
wl_type	char(12)	This row contains the ready queue size, user CPU time, and system CPU time
wl_workload_1	float	Most recent workload activity
wl_workload_2	float	Second most recent workload activity
wl_workload_3	float	Third most recent workload activity
wl_workload_4	float	Fourth most recent workload activity
wl_workload_5	float	Fifth most recent workload activity
wl_workload_6	float	Sixth most recent workload activity
wl_workload_7	float	Seventh most recent workload activity
wl_workload_8	float	Eighth most recent workload activity
wl_workload_9	float	Ninth most recent workload activity
wl_workload_10	float	Tenth most recent workload activity
wl_workload_11	float	Eleventh most recent workload activity
wl_workload_12	float	Twelfth most recent workload activity
wl_workload_13	float	Thirteenth most recent workload activity
wl_workload_14	float	Fourteenth most recent workload activity

Table 90. sysha_workload table information

(continued)

Column	Type	Description
wl_workload_15	float	Fifteenth most recent workload activity
wl_workload_16	float	Sixteenth most recent workload activity
wl_workload_17	float	Seventeenth most recent workload activity
wl_workload_18	float	Eighteenth most recent workload activity
wl_workload_19	float	Nineteenth most recent workload activity
wl_workload_20	float	Twentieth most recent workload activity

sysipl

The **sysipl** table provides information about the status of index page logging at the primary server.

Table 91. sysipl table information

Column	Type	Description
ipl_status	integer	Index page logging status
ipl_time	integer	Time at which index page logging was enabled

syslocks

The **syslocks** table provides information about all the currently active locks in the database server.

Table 92. syslocks table information

Column	Type	Description
dbsname	char(128)	Database name
tabname	char(128)	Table name
rowidlk	integer	Real rowid, if it is an index key lock
keynum	smallint	Key number of index key lock

Table 92. syslocks table information

(continued)

<table>
<tr><th>Column</th><th>Type</th><th colspan="2">Description</th></tr>
<tr><td rowspan="10">type</td><td rowspan="10">char(4)</td><td colspan="2">Type of lock</td></tr>
<tr><td>B</td><td>Byte lock</td></tr>
<tr><td>IS</td><td>Intent shared lock</td></tr>
<tr><td>S</td><td>Shared lock</td></tr>
<tr><td>XS</td><td>Shared key value held by a repeatable reader</td></tr>
<tr><td>U</td><td>Update lock</td></tr>
<tr><td>IX</td><td>Intent exclusive lock</td></tr>
<tr><td>SIX</td><td>Shared intent exclusive lock</td></tr>
<tr><td>X</td><td>Exclusive lock</td></tr>
<tr><td>XR</td><td>Exclusive key value held by a repeatable reader</td></tr>
<tr><td>owner</td><td>integer</td><td colspan="2">Session ID of the lock owner</td></tr>
<tr><td>waiter</td><td>integer</td><td colspan="2">Session ID of the user waiting for the lock. If more than one user is waiting, only the first session ID appears.</td></tr>
</table>

syslogs

The **syslogs** table provides information about space use in logical-log files. In the **flags** column, each bit position represents a separate flag. For example, for a log file, the **flags** column can have flags set for both current log file and temporary log file. Thus, it might be easier to read values in the **flags** column if the values are returned using the HEX function.

Table 93. syslogs table information

Column	Type	Description
number	smallint	Logical-log file number
uniqid	integer	Log-file ID
size	integer	Number of pages in the log file
used	integer	Number of pages used in the log file
is_used	integer	1 If file is used, 0 if not

Table 93. syslogs table information

(continued)

<table>
<tr><th>Column</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td>is_current</td><td>integer</td><td colspan="3">1 If file is the current file, 0 if not</td></tr>
<tr><td>is_backed_up</td><td>integer</td><td colspan="3">1 If file has been backed up, 0 if not</td></tr>
<tr><td>is_new</td><td>integer</td><td colspan="3">1 If the log has been added since the last level-0 dbspace backup, 0 if not</td></tr>
<tr><td>is_archived</td><td>integer</td><td colspan="3">1 If file has been placed on the backup tape, 0 if not</td></tr>
<tr><td>is_temp</td><td>integer</td><td colspan="3">1 If the file is flagged as a temporary log file, 0 if not</td></tr>
<tr><td rowspan="7">flags</td><td rowspan="7">smallint</td><td>Flags</td><td>Hexadecimal</td><td>Meaning</td></tr>
<tr><td>1</td><td>0x01</td><td>Log file is in use</td></tr>
<tr><td>2</td><td>0x02</td><td>File is current log file</td></tr>
<tr><td>4</td><td>0x04</td><td>Log file has been backed up</td></tr>
<tr><td>8</td><td>0x08</td><td>File is newly added log file</td></tr>
<tr><td>16</td><td>0x10</td><td>Log file has been written to dbspace backup media</td></tr>
<tr><td>32</td><td>0x20</td><td>Log is a temporary log file</td></tr>
</table>

syslogfil table

The syslogfil table provides information about the logical log files.

Table 94. Information about the columns in the syslogfil table.

Column	Type	Description
address	int8	Memory address of the logfile structure
number	small integer	Log file number
flags	integer	For a description of the values and their meanings, see the **Flag values** section below.
fillstamp	integer	Internal timestamp when the log file was filled
filltime	integer	UNIX™ time when the log file was filled
uniqid	integer	Unique ID for the log file
chunk	integer	Number of the chunk that contains the log file
offset	integer	Page offset in the chunk where log file begins
size	integer	Total number of pages in the log file

Table 94. Information about the columns in the syslogfil table. (continued)

Column	Type	Description
used	integer	Number of pages used in the log file

Flag values

The flag values correspond to many of the flag values for the onstat -l command.

Hexadecimal	Onstat -l flag value	Meaning
0x1	U	Log file is in use
0x2	C	File is current log file
0x4	B	Log file has been backed up
0x8	A	File is a newly added log file
0x20	None	A temporary log file
0x40	D	Log file will be dropped after the file is archived
0x4000	L	Log file contains the last checkpoint written

sysmgminfo

The **sysmgminfo** table provides an overview of the Memory Grant Manager (MGM) and Parallel Data Query (PDQ) information.

Table 95. sysmgminfo table information

Column	Type	Description
max_query	integer	Maximum number of active queries allowed
total_mem	integer	Total MGM memory
avail_mem	integer	Free MGM memory
total_seq	integer	Total number of sequential scans
avail_seq	integer	Unused sequential scans
active	integer	Number of active MGM queries
ready	integer	Number of ready MGM queries

Table 95. sysmgminfo table information

(continued)

Column	Type	Description
min_free_mem	integer	Minimum free MGM memory
avg_free_mem	float	Average free MGM memory
std_free_mem	float	Standard free MGM memory
min_free_seq	integer	Minimum free MGM sequential scans
avg_free_seq	float	Average free MGM sequential scans
std_free seq	float	Standard free MGM sequential scans
max_active	integer	Maximum active MGM SQL operations
cnt_active	integer	Number of active MGM SQL operations
avg_active	float	Average active MGM SQL operations
std_active	float	Standard active MGM SQL operations
max_ready	integer	Maximum ready MGM SQL operations
cnt_ready	integer	Number of ready MGM SQL operations
avg_ready	float	Average ready MGM SQL operations
std_ready	float	Standard ready MGM SQL operations

sysnetclienttype

The **sysnetclienttype** table provides an overview of the network activity for each client type.

Column	Type	Description
nc_cons_allowed	integer	Whether or not connections are allowed
nc_accepted	integer	Number of connections that were accepted
nc_rejected	integer	Number of network connections that were rejected
nc_reads	int8	Number of network reads for this client type
nc_writes	int8	Number of network writes for this client type
nc_name	char(18)	Name of the client type

sysnetglobal

The **sysnetglobal** table provides an overview of the system network.

Column	Type	Description
ng_reads	int8	Number of network reads
ng_writes	int8	Number of network writes
ng_connects	int8	Number of network connections
ng_his_read_count	int8	Number of network reads by users who have disconnected **ng_his_read_bytes**
ng_his_read_bytes	int8	Data transferred to the server by users who have disconnected
ng_his_write_count	int8	Number of network writes by users who have disconnected
ng_his_write_bytes	int8	Data transferred to the client by users who have disconnected
ng_num_netscbs	integer	Number of network subscribers
ng_max_netscbs	integer	Maximum number of network subscribers
ng_free_thres	integer	Threshold for the maximum number of freed buffers in the buffer list
ng_free_cnt	integer	Number of times the ng_free_thres limit has been reached
ng_wait_thres	integer	Threshold for the maximum number of buffers that can be held in the buffer list for one connection
ng_wait_cnt	integer	Number of times the ng_wait_thres limit has been reached
ng_pvt_thres	integer	Threshold for the maximum number of freed buffers in the private buffer queue
ng_netbuf_size	integer	Size of the transport network buffers
ng_buf_alloc	integer	Number of network buffers allocated
ng_buf_alloc_max	integer	Maximum value of allocated network buffers
ng_netscb_id	integer	Next netscb id

sysnetworkio table

The **sysnetworkio** table contains information about the system network.

Column	Type	Description
net_id	integer	Netscb id
sid	integer	Session id

Column	Type	Description
net_netscb	int8	Netscb prt
net_client_type	integer	Client type Int
net_client_name	char(12)	Client protocol name
net_read_cnt	int8	Number of network reads
net_write_cnt	int8	Number of network writes
net_open_time	integer	Time this session connected
net_last_read	integer	Time of the last read from the network
net_last_write	integer	Time of the last write from the network
net_stage	integer	Connect / Disconnect / Receive
net_options	integer	Options from sqlhosts
net_protocol	integer	Protocol
net_type	char(10)	Type of network protocol
net_server_fd	integer	Server fd
net_poll_thread	integer	Poll thread

sysonlinelog

The **sysonlinelog** table provides a view of the information stored in the online.log file.

Column	Type	Description
offset	int8	File offset
next_offset	int8	Offset to the next message
line	char(4096)	Single line of text from the file

sysprofile

The **sysprofile** table contains profile information about the database server.

Column	Type	Description
name	char(13)	Name of profiled event. (See table that follows for a list of possible events.)
value	integer	Value of profiled event. (See table that follows for a list of possible events.)

The following table lists the events that, together with a corresponding value, make up the rows of the **sysprofile** table.

Events Profiled in sysprofile	Description
dskreads	Number of actual reads from disk
bufreads	Number of reads from shared memory
dskwrites	Actual number of writes to disk
bufwrites	Number of writes to shared memory
isamtot	Total number of calls
isopens	isopen calls
isstarts	isstart calls
isreads	isread calls
iswrites	iswrite calls
isrewrites	isrewrite calls
isdeletes	isdelete calls
iscommits	iscommit calls
isrollbacks	isrollback calls
ovlock	Overflow lock table
ovuser	Overflow user table
ovtrans	Overflow transaction table
latchwts	Latch request waits
bufwts	Buffer waits
lockreqs	Lock requests
lockwts	Lock waits
ckptwts	Checkpoint waits
deadlks	Deadlocks
lktouts	Deadlock time-outs
numckpts	Number checkpoints
plgpagewrites	Physical-log pages written
plgwrites	Physical-log writes
llgrecs	Logical-log records
llgpagewrites	Logical-log writes

Events Profiled in sysprofile	Description
llgwrites	Logical-log pages written
pagreads	Page reads
pagwrites	Page writes
flushes	Buffer-pool flushes
compress	Page compresses
fgwrites	Foreground writes
lruwrites	Least-recently used (LRU) writes
chunkwrites	Writes during a checkpoint
btradata	Read-ahead data pages read through index leaf node
btraidx	Read-ahead data pages read through index branch or root node
dpra	Data pages read into memory with read-ahead feature
rapgs_used	Read-ahead data pages that user used
seqscans	Sequential scans
totalsorts	Total sorts
memsorts	Sorts that fit in memory
disksorts	Sorts that did not fit in memory
maxsortspace	Maximum disk space used by a sort

sysproxyagents

The **sysproxyagents** table contains information about all proxy agent threads. Proxy agent threads run on the primary server and accept requests from secondary servers to process DML operations. The primary server also contains a proxy distributor that handles secondary server updates. Secondary servers determine how many instances of the proxy distributor to create based on the UPDATABLE_SECONDARY setting in the secondary server's ONCONFIG file.

Column	Type	Description
tid	integer	Transaction ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the secondary server session.
flags	integer	Flags of the proxy agent thread.
proxy_id	integer	ID of the proxy distributor on behalf of the currently executing proxy agent thread (TID).

Column	Type	Description
source_session_id	integer	ID of the user's session on the secondary server.
proxy_txn_id	integer	Number of the current transaction. These numbers are unique to the proxy distributor.
current_seq	integer	The sequence number of the current operation in the current transaction.
sqlerrno	integer	Error number of any SQL error (or 0 on success)
iserrno	integer	Error number of any ISAM/RSAM error (or 0 on success)

sysproxydistributors

The **sysproxydistributors** table contains information about the proxy distributors.

On the primary server, this table contains information about all of the proxy distributors in a high-availability cluster. On a secondary server, this table contains information about only those proxy distributors that are assigned to process updates to the secondary server.

Column	Type	Description
node_name	char	Name of the secondary server as it is known by the primary server (for example, ONEDB_SERVER, HA_ALIAS, and so on).
proxy_id	integer	ID of the proxy distributor. These IDs are unique within a high-availability cluster.
transaction_count	integer	Number of transactions currently being processed by the proxy distributor.
hot_row_total	integer	Total number of hot rows ever handled by the proxy distributor. A hot row is a row on a secondary server that is updated multiple times by more than one client. When a row is updated multiple times, the secondary server reads the before image from the primary server by placing an update lock on the row if the most recent update operation from a different session is not replayed on the secondary server.

sysproxysessions table

The **sysproxysessions** table contains information about each of the sessions that are using redirected-write functionality. This table is only valid on the secondary server.

The following table provides information about the columns in the **sysproxysessions** table:

Column	Type	Description
session_id	integer	ID of a user's session on the secondary server.

Column	Type	Description
proxy_id	integer	ID of the proxy distributor on behalf of which the proxy agent thread (TID) is running
proxy_tid	integer	Transaction ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the secondary server session.
proxy_txn_id	integer	Number of the current transaction. These numbers are unique to the proxy distributor.
current_seq	integer	The sequence number of the current operation in the current transaction.
pending_ops	integer	The number of operations buffered on the secondary server that have not yet been sent to the primary server.
reference_count	integer	Indicates the number of threads (for example, sqlexec, sync reply, recovery, and so on) that are using the information for this transaction. When reference_count equals 0, the transaction processing has completed (either successfully or unsuccessfully).

sysproxytxnops table

The **sysproxytxnops** table contains information about each of the transactions that are running through each proxy distributor.

On the primary server, this table contains information about all of the proxy distributors in the high-availability cluster. On a secondary server, this table only contains information about the proxy distributors used to process updates to the secondary server.

The following table provides information about the columns in the **sysproxytxnops** table:

Column	Type	Description
proxy_id	integer	ID of the proxy distributor. These IDs are unique within a high-availability cluster.
proxy_txn_id	integer	Number of the transaction. These numbers are unique to the proxy distributor.
sequence_number	integer	The number of the operation.
operation_type	char(10)	The type of operation to be performed; Insert, Update, Delete, or Other.
rowidn	integer	The ID of the row on which to apply the operation.
table	char	The full table name, trimmed to fit a reasonable length. Format: *database*:*owner*.*tablename*
sqlerrno	integer	Error number of any SQL error (or 0 on success)

sysproxytxns table

The **sysproxytxns** table contains information about all of the current transactions that are running through each proxy distributor.

On the primary server, this table contains information about each of the proxy distributors in the high-availability cluster. On a secondary server, this table only contains information about the proxy distributors used to process updates to the secondary server.

The following table provides information about the columns in the **sysproxytxns** table:

Column	Type	Description
proxy_id	integer	ID of the proxy distributor. These IDs are unique within a high-availability cluster.
proxy_txn_id	integer	Number of the transaction. These numbers are unique to the proxy distributor.
reference_count	integer	Indicates the number of threads (for example, sqlexec, sync reply, recovery, and so on) that are using the information for this transaction. When the count becomes 0 this indicates the transaction processing is complete. (either successfully or unsuccessfully).
pending_ops	integer	On the primary server: the number of operations received from the secondary server that have not yet been processed. On the secondary server, the number of operations buffered on the secondary server that have not yet been sent to the primary server.
proxy_sid	integer	Proxy Session ID

sysptnhdr

The **sysptrhdr** table contains information about partition headers.

Column	Type	Description
partnum	integer	Partnum of the table
flags	integer	Partition flags
rowsize	integer	Row size (maximum for variable)
ncols	smallint	Number of VARCHAR or BLOB columns
nkeys	smallint	Number of indexes
nextns	smallint	Number of extents
pagesize	smallint	Page size
created	integer	Date created
serialv	integer	Current SERIAL value

Column	Type	Description
fextsiz	integer	First extent size, in pages
nextsiz	integer	Next extent size, in pages
nptotal	integer	Number of pages allocated
npused	integer	Number of pages used
npdata	integer	Number of data pages
octptnm	integer	Optical BLOB partnum
lockid	integer	Table lock ID
nrows	bigint	Number of data rows
ninserts	bigint	Number of insert operations
nupdates	bigint	Number of update operations
ndeletes	bigint	Number of delete operations
cur_serial8	int8	Current SERIAL8 value
cur_bigserial	bigint	Current BIGSERIAL value
dbsnum	integer	Number of partitions in the dbspace
pta_oldvers	smallint	In-place alter
pta_newvers	smallint	In-place alter
pta_bmpagenum	integer	In-place alter
pta_totpgs	integer	In-place alter
pta_opems_allocd	integer	In-place alter
pta_opems_filled	integer	In-place alter
glscollname	char(32)	In-place alter
flags2	integer	Partition flags2
sid	integer	Temporary table session ID

You can run the following query to see the number of allocated pages for temporary tables:

```
SELECT i.sid, hex(i.flags) flags, hex(i.partnum) partition,
   trim(n.dbsname) || ":" || trim(n.owner) || ":" || trim(n.tabname) table,
   i.nptotal allocated_pages
  FROM sysmaster:systabnames n, sysmaster:sysptnhdr i
 WHERE (sysmaster:bitval(i.flags, "0x0020") = 1)
   AND i.partnum = n.partnum
```

For example, the query can return information similar to the following output:

```
session with query "select * from customer into temp good "
sid              60
flags            0x00000861
partition        0x00100249
table            demo:informix:good
allocated_pages  8

session with temp table generated from query "select from <view>"
sid              64
flags            0x00008821
partition        0x00100249
table            demo:informix:_temptable
allocated_pages  8

temp table from sorting
sid              33
flags            0x000048A0
partition        0x00200004
table            SORTTEMP:informix:th_tmprun_0x4a1b2370
allocated_pages  128

temp table from hashing
sid              31
flags            0x000048A0
partition        0x00200003
table            HASHTEMP:informix:th_overflow_0xffffffffffffffff
allocated_pages  16
```

sysptprof table

The **sysptprof** table lists information about a tblspace. Tblspaces correspond to tables.

Profile information for a table is available only when a table is open. When the last user who has a table open closes it, the tblspace in shared memory is freed, and any profile statistics are lost.

The following table provides information about the columns in the **sysptprof** table:

Column	Type	Description
dbsname	char(128)	Database name
tabname	char(128)	Table name
partnum	integer	Partition (tblspace) number
lockreqs	integer	Number of lock requests
lockwts	integer	Number of lock waits
deadlks	integer	Number of deadlocks
lktouts	integer	Number of lock timeouts
isreads	integer	Number of isreads
iswrites	integer	Number of iswrites

Column	Type	Description
isrewrites	integer	Number of isrewrites
isdeletes	integer	Number of isdeletes
bufreads	integer	Number of buffer reads
bufwrites	integer	Number of buffer writes
seqscans	integer	Number of sequential scans
pagreads	integer	Number of page reads
pagwrites	integer	Number of page writes

sysrepevtreg table

Use the **sysrepevtreg** pseudo table to register for a pre-defined set of events from the Connection Manager.

The following table provides information about the columns in the **sysrepevtreg** table:

Table 96. sysrepevtreg table information

Column	Type	Description
evt_bitmap	integer	Event ID bitmap
evt_timeout	integer	Maximum time in seconds that the client can wait for event data. Valid timeout values are: • 0; no wait (default) • -1; wait forever • *n* (where *n* > 0) wait *n* seconds
evt_hwm	integer	Pending event list high-water mark
evt_info	char(256)	Event information (Not yet implemented)

sysrepstats table

Use the **sysrepstats** table to post events to Connection Manager.

The following table provides information about the columns in the **sysrepstats** table:

Table 97. sysrepstats table information

Column	Type	Description
repstats_type	integer	Event ID
repstats_subtype	integer	Sub event ID
repstats_time	integer	Time at which event was initiated
repstats_ver	integer	Version number of event data
repstats_desc	lvarchar	Event data

User Interface for sysrepstats and sysrepevtreg Tables

Client applications can post events to Connection Manager or to other clients by inserting event information into the **sysrepstats** pseudo table. Client applications can register events using the sysmaster pseudo table **sysrepevtreg**, and receive event data by issuing select or fetch statements against the **sysrepstats** pseudo table.

By posting events to the **sysrepstats**, you can issue control messages to Connection Manager without having to directly connect to Connection Manager itself.

When Connection Manager registers that it wishes to receive events, it passes a bitmap of the event types that it wants to receive. As events are received, they are posted to the thread that placed the request.

Event Classes

The following table lists each event class, its bit value, and a description of the event class.

Table 98. Event Classes

Event class name	Bit value	Description
REPEVT_CLUST_CHG	0x1	Event class for High-Availability cluster changes
REPEVT_CLUST_PERFSTAT	0x2	Event class for workload statistics for the server nodes in a High-Availability cluster
REPEVT_CLUST_LATSTAT	0x4	Event class for replication latency information for server nodes in a High-Availability cluster
REPEVT_CM_ADM	0x8	Connection Manager administration commands
REPEVT_SRV_ADM	0x10	Event class for server mode changes

Table 98. Event Classes

(continued)

Event class name	Bit value	Description
REPEVT_ER_ADM	0x20	Event class for events related to Enterprise Replication (ER)
REPEVT_CLIENT	0x40	User-defined client event

Sub-events for the Event Class REPEVT_CLUST_CHG

The following table lists sub-events for the event class REPEVT_CLUST_CHG:

Table 99. Sub-events for the Event Class REPEVT_CLUST_CHG

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_CLUST_ADD	1	Adding new node to a High-Availability cluster	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_DROP	2	Dropping a node from a High-Availability cluster	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_CON	3	High-Availability secondary node connected to primary server	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_DIS	4	High-Availability secondary node disconnected from primary server	Only at primary server in a High-Availability cluster
REPEVT_SUB_CLUST_NEWPRIM	5	High-Availability primary node changed	Only at secondary servers in a High-Availability cluster
REPEVT_SUB_CLUST_DROFF	6	HDR secondary node disconnected from primary server	HDR primary and secondary servers
REPEVT_SUB_CLUST_DRON	7	HDR secondary node connected to primary server	HDR primary and secondary servers

Sub-events for the Event Class REPEVT_CLUST_PERFSTAT

The following table lists sub-events for the event class REPEVT_CLUST_PERFSTAT:

Table 100. Sub-events for the Event Class REPEVT_CLUST_PERFSTAT

Sub-event name	Value	Description	Event available at:
REPEVT_ SUB_LOCAL_PERFSTAT	1	Work load statistics for local server	All servers in a High-Availability cluster
REPEVT_ SUB_REMOTE_PERFSTAT	2	Work load statistics for High-Availability secondary servers	Only at the primary server in a High-Availability cluster

Sub-events for the Event Class REPEVT_CLUST_LATSTAT

The following table lists sub-events for the event class REPEVT_CLUST_LATSTAT:

Table 101. Sub-events for the Event Class REPEVT_CLUST_LATSTAT

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_LOCAL_LATSTAT	1	Replication latency statistics for secondary servers in a High-Availability cluster	Only at the primary server in a High-Availability cluster

Sub-events for the Event Class REPEVT_CM_ADM

The following table lists sub-events for the event class REPEVT_CM_ADM:

Table 102. Sub-events for the Event Class REPEVT_CM_ADM

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_CM_ADM_REQ	1	Command request	All HCL OneDB™ server instances
REPEVT_SUB_CM_ADM_ACK	2	Command response	All HCL OneDB™ server instances
REPEVT_SUB_CM_REG	3	Connection Manager registered with server	All HCL OneDB™ server instances

Table 102. Sub-events for the Event Class REPEVT_CM_ADM

(continued)

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_CM_DEREG	4	Connection Manager de-registered with server	All HCL OneDB™ server instances
REPEVT_SUB_CM_FATAL	5	Connection Manager terminated without de-registering with server	All HCL OneDB™ server instances

Sub-events for the Event Class REPEVT_SRV_ADM

The following table lists sub-events for the event class REPEVT_SRV_ADM:

Table 103. Sub-events for the Event Class REPEVT_SRV_ADM

Sub-event name	Value	Description	Event available at:
REPEVT_ SUB_SRV_BLK	1	Server blocked due to DDRBLOCK	All HCL OneDB™ server instances
REPEVT_ SUB_SRV_UBLK	2	Server unblocked; DDRBLOCK removed	All HCL OneDB™ server instances

Sub-events for the Event Class REPEVT_ER_ADM

The following table lists sub-events for the event class REPEVT_ER_ADM:

Table 104. Sub-events for the Event Class REPEVT_ER_ADM

Sub-event name	Value	Description	Event available at:
REPEVT_SUB_ER_SPOOL_FULL	1	ER blocked while waiting for space to be added in either the queue data sbspace or dbspace, or in the grouper paging sbspace.	Enterprise Replication server nodes

sysrsslog

The **sysrsslog** table captures information about RS secondary servers at the primary server.

Table 105. sysrsslog table information

Column	Type	Description
server_name	char(128)	Server name
from_cache	integer	Total pages read from log buffer cache
from_disk	integer	Total pages read from disk
logpages_tossed	integer	Total number of log pages not written to log buffer cache

sysscblst

These columns of the **sysscblst** table provide information about session memory amounts.

Column	Type	Description
memtotal	integer	Total memory available
memused	integer	Total memory used

sysscelem

The **syssscelem** table provides information about Statement Cache Entries.

Column	Type	Description
uniqid	integer	Unique id of element
lru	integer	Index of lru queue
hash	integer	Hash value of cached entry
ref_cnt	integer	Num threads referencing the statement
hits	integer	Num times element used
flag	integer	• Invalid 0x1 • Fully Cached 0x2 • Inserting 0x4
valid	integer	Valid 1, Invalid 0

Column	Type	Description
locked	integer	Locked 1, Unlocked 0
heap_ptr	bigint	Address of memory heap for cache entry
database	char(128)	Database name
user	char(32)	User
stmtstring	char(16000)	Statement string
queryplan	char(16000)	Query plan

syssesappinfo

The **syssesappinfo** table in the **sysmaster** displays information on Distributed Relational Database Architecture™ (DRDA®) client sessions. The **syssesappinfo** table has the following columns.

Table 106. syssesappinfo table column information

Column	Type	Explanation
sesapp_sid	INTEGER	Client session ID
sesapp_name	CHAR(128)	Session application name
sesapp_value	CHAR(512)	Session value

syssesprof

The **syssesprof** table lists cumulative counts of the number of occurrences of user actions such as writes, deletes, or commits.

Column	Type	Description
sid	integer	Session ID
lockreqs	integer	Number of locks requested
locksheld	integer	Number of locks currently held
lockwts	integer	Number of times waited for a lock
deadlks	integer	Number of deadlocks detected
lktouts	smallint	Number of deadlock time-outs
logrecs	integer	Number of logical-log records written
isreads	integer	Number of reads

Column	Type	Description
iswrites	integer	Number of writes
isrewrites	integer	Number of rewrites
isdeletes	integer	Number of deletes
iscommits	integer	Number of commits
isrollbacks	integer	Number of rollbacks
longtxs	integer	Number of long transactions
bufreads	integer	Number of buffer reads
bufwrites	integer	Number of buffer writes
seqscans	integer	Number of sequential scans
pagreads	integer	Number of page reads
pagwrites	integer	Number of page writes
total_sorts	integer	Number of total sorts
dsksorts	integer	Number of sorts that did not fit in memory
max_sortdiskspace	integer	Maximum space used by a sort
logspused	integer	Number of bytes of logical-log space used by current transaction of session
maxlogsp	integer	Maximum number of bytes of logical-log usage for any single transaction since the session started

syssessions

The **syssessions** table provides general information on each user connected to the database server. In the **state** column, each bit position represents a separate flag. Thus, it might be easier to read values in the **state** column if the values are returned using the HEX function.

Table 107. syssessions table information

Column	Type	Description
sid	integer	Session ID
username	char(32)	User ID
uid	smallint	User ID number

Table 107. syssessions table information

(continued)

<table>
<tr><th>Column</th><th>Type</th><th>Description</th></tr>
<tr><td>pid</td><td>integer</td><td>Process ID of the client</td></tr>
<tr><td>hostname</td><td>char(256)</td><td>Hostname of client</td></tr>
<tr><td>tty</td><td>char(16)</td><td>Name of the user's stderr file</td></tr>
<tr><td>connected</td><td>integer</td><td>Time that user connected to the database server</td></tr>
<tr><td>feprogram</td><td>char(255)</td><td>Absolute path of the executable program or application</td></tr>
<tr><td>pooladdr</td><td>integer</td><td>Session pool address</td></tr>
<tr><td>is_wlatch</td><td>integer</td><td>1 if the primary thread for the session is waiting for a latch</td></tr>
<tr><td>is_wlock</td><td>integer</td><td>1 if the primary thread for the session is waiting for a lock</td></tr>
<tr><td>is_wbuff</td><td>integer</td><td>1 if the primary thread for the session is waiting for a buffer</td></tr>
<tr><td>is_wckpt</td><td>integer</td><td>1 if the primary thread for the session is waiting for a checkpoint</td></tr>
<tr><td>is_wlogbuf</td><td>integer</td><td>1 if the primary thread for the session is waiting for a log buffer</td></tr>
<tr><td>is_wtrans</td><td>integer</td><td>1 if the primary thread for the session is waiting for a transaction</td></tr>
<tr><td>is_monitor</td><td>integer</td><td>1 if the session is a special monitoring process</td></tr>
<tr><td>is_incrit</td><td>integer</td><td>1 if the primary thread for the session is in a critical section</td></tr>
<tr><td>state</td><td>integer</td><td>
<table>
<tr><td>Flags</td><td>Hexadecimal</td><td>Meaning</td></tr>
<tr><td>1</td><td>0x00000001</td><td>User structure in use</td></tr>
<tr><td>2</td><td>0x00000002</td><td>Waiting for a latch</td></tr>
<tr><td>4</td><td>0x00000004</td><td>Waiting for a lock</td></tr>
<tr><td>8</td><td>0x00000008</td><td>Waiting for a buffer</td></tr>
<tr><td>16</td><td>0x00000010</td><td>Waiting for a checkpoint</td></tr>
<tr><td>32</td><td>0x00000020</td><td>In a read call</td></tr>
<tr><td>64</td><td>0x00000040</td><td>Writing logical-log file to backup tape</td></tr>
<tr><td>128</td><td>0x00000080</td><td>ON-Monitor (UNIX™)</td></tr>
<tr><td>256</td><td>0x00000100</td><td>In a critical section</td></tr>
<tr><td>512</td><td>0x00000200</td><td>Special daemon</td></tr>
</table>
</td></tr>
</table>

Table 107. syssessions table information

(continued)

Column	Type	Description		
		1024	0x00000400	Archiving
		2048	0x00000800	Clean up dead processes
		4096	0x00001000	Waiting for write of log buffer
		8192	0x00002000	Special buffer-flushing thread
		16384	0x00004000	Remote database server
		32768	0x00008000	Deadlock timeout used to set RS_timeout
		65536	0x00010000	Regular lock timeout
		262144	0x00040000	Waiting for a transaction
		524288	0x00080000	Primary thread for a session
		1048576	0x00100000	Thread for building indexes
		2097152	0x00200000	B-tree cleaner thread

syssessiontempspaceusage

The **syssessiontempspaceusage** table contains information about each session's temp space usage, includes both implicit and explicit temp tables.

Column	Type	Description
sid	integer	ID of the session which allocated temp space
flags	char(10)	Partition flags of the temp space partition
partition	char(10)	Partition number of the temp space partition
table	lvarchar(290)	Table name of the temp space partition
allocated_pages	integer	Number of pages allocated by the temp space partition

Example:

```
select * from syssessiontempspaceusage

sid              53
flags            0x00000861
partition        0x001001A4
table            stores_demo:userabc:foo
allocated_pages  8
```

syssmx

The **syssmx** table provides SMX (server multiplexer group) connection information.

Table 108. syssmx table column information

Column	Type	Description
address	int8	SMX pipe address
name	char(128)	Target server name
encryption_status	char(20)	Enabled or disabledReserved for future use
buffers_sent	integer	Number of buffers sent
buffers_recv	integer	Number of buffers received
bytes_sent	int8	Number of bytes sent
bytes_recv	int8	Number of bytes received
reads	integer	Number of read calls
writes	integer	Number of write calls
retries	integer	Number of write call retries

syssmxses

The **syssmxses** table provides SMX (server multiplexer group) session information.

Table 109. syssmxses table column information

Column	Type	Description
name	char(128)	Target server name
address	int8	SMX session address
client_type	char(20)	SMX client type
reads	integer	Number of read calls
writes	integer	Number of write calls

syssqexplain table

The **syssqexplain** pseudo table stores information about SQL queries.

The information stored includes the plan of the query optimizer, an estimate of the number of rows returned, and the relative cost of the query.

Table 110. The syssqexplain pseudo table

Column	Type	Description
sqx_sessionid	INTEGER	The session ID associated with the SQL statement.
sqx_sdbno	INTEGER	The position of the query in the array of session IDs.
sqx_iscurrent	CHAR	Whether the query is the current SQL statement.
sqx_executions	INTEGER	The total number of executions of the query.
sqx_cumtime	FLOAT	The cumulative time to run the query. **Important:** If SQL tracing is disabled a zero is shown.
sqx_bufreads	INTEGER	The number of buffer reads performed while running the query. **Important:** If SQL tracing is disabled a zero is shown.
sqx_pagereads	INTEGER	The number of page reads performed while running the query. **Important:** If SQL tracing is disabled a zero is shown.
sqx_bufwrites	INTEGER	The number of buffer writes performed while running the query. **Important:** If SQL tracing is disabled a zero is shown.
sqx_pagewrites	INTEGER	The number of page writes performed while running the query. **Important:** If SQL tracing is disabled a zero is shown.
sqx_totsorts	INTEGER	The total number of sorts performed while running the query. **Important:** If SQL tracing is disabled a zero is shown.
sqx_dsksorts	INTEGER	The number of disk sorts performed while running the query.

Table 110. The syssqexplain pseudo table (continued)

Column	Type	Description
		Important: If SQL tracing is disabled a zero is shown.
sqx_sortspmax	INTEGER	The maximum disk space required by a sort.
sqx_conbno	SMALLINT	The position in the conblock list.
sqx_ismain	CHAR	Whether the query is in the main block for the statement.
sqx_selflag	VARCHAR(200,0)	The type of SQL statement, for example: SELECT, UPDATE, DELETE.
sqx_estcost	INTEGER	The estimated cost of the query.
sqx_estrows	INTEGER	The estimated number of rows returned by the query.
sqx_seqscan	SMALLINT	The number of sequential scans used by the query.
sqx_srtscan	SMALLINT	The number of sort scans used by the query.
sqx_autoindex	SMALLINT	The number of autoindex scans used by the query.
sqx_index	SMALLINT	The number of index paths used by the query.
sqx_remsql	SMALLINT	The number of remote paths used by the query.
sqx_mrgjoin	SMALLINT	The number of sort-merge joins used by the query.
sqx_dynhashjoin	SMALLINT	The number of dynamic hash joins used by the query.
sqx_keyonly	SMALLINT	The number of key-only scans used by the query.
sqx_tempfile	SMALLINT	The number of temporary files used by the query.
sqx_tempview	SMALLINT	The number of temporary tables for views created by the query.
sqx_secthreads	SMALLINT	The number of secondary threads used by the query.
sqx_sqlstatement	CHAR	The SQL query that was run.

syssqltrace

The **syssqltrace** table provides detailed information about a single SQL statement.

Column	Type	Description
sql_id	int8	Unique SQL execution ID
sql_address	int8	Address of the statement in the code block
sql_sid	int	Database session ID of the user running the SQL statement
sql_uid	int	User ID of the statement running the SQL

Column	Type	Description
sql_stmttype	int	Statement type
sql_stmtname	char(40)	Statement type displayed as a word
sql_finishtime	int	Time this statement completed (UNIX™)
sql_begintxtime	int	Time this transaction started
sql_runtime	float	Statement execution time
sql_pgreads	int	Number of disk reads for this SQL statement
sql_bfreads	int	Number of buffer reads for this SQL statement
sql_rdcache	float	Percentage of time the page was read from the buffer pool
sql_bfidxreads	int	Number of index page buffer reads
sql_pgwrites	int	Number of pages written to disk
sql_bfwrites	int	Number of pages modified and returned to the buffer pool
sql_wrcache	float	Percentage of time a page was written to the buffer pool but not to disk
sql_lockreq	int	Total number of locks required by this SQL statement
sql_lockwaits	int	Number of times the SQL statement waited on locks
sql_lockwttime	float	Time the system waited for locks during SQL statement
sql_logspace	int	Amount of space the SQL statement used in the logical log
sql_sorttotal	int	Number of sorts that ran for the statement
sql_sortdisk	int	Number of sorts that ran on disk
sql_sortmem	int	Number of sorts that ran in memory
sql_executions	int	Number of times the SQL statement ran
sql_totaltime	float	Total amount of time spent running the statement
sql_avgtime	float	Average amount of time spent running the statement
sql_maxtime	float	Maximum amount of time spent executing the SQL statement
sql_numiowaits	int	Number of times an I/O operation had to wait
sql_avgiowaits	float	Average amount of time that the SQL statement had to wait
sql_totaliowaits	float	Total amount of time that the SQL statement had to wait for I/O. This excludes any asynchronous I/O.
sql_rowspersec	float	Average number of rows (per second) produced
sql_estcost	int	Cost associated with the SQL statement

Column	Type	Description
sql_estrows	int	Estimated number of rows returned for the SQL statement as predicted by the optimizer
sql_actualrows	int	Number of rows returned for the SQL statement
sql_sqlerror	int	SQL error number
sql_isamerror	int	RSAM/ISAM error number
sql_isollevel	int	Isolation level of the SQL statement.
sql_sqlmemory	int	Number of bytes needed to execute the SQL statement
sql_numiterators	int	Number of iterators used by the statement
sql_database	char(128)	Database name
sql_numtables	int	Number of tables used in executing the SQL statement
sql_tablelist	char(4096)	List of table names directly referenced in the SQL statement. If the SQL statement fires triggers that execute statements against other tables, the other tables are not listed.
sql_statement	char(1600)	SQL statement that ran

syssqltrace_hvar

The **syssqltrace_hvar** table describes information about the SQL tracing host variable.

Column	Type	Description
sql_id	int8	SQL execution ID
sql_address	int8	Address of the SQL statement block
sql_hvar_id	int	ID of the SQL host variable
sql_hvar_flags	int	Flags for the host variable
sql_hvar_typeid	int	Type ID of the host variable
sql_hvar_xtypeid	int	xtype ID of the host variable
sql_hvar_ind	int	Index of the host variable
sql_hvar_type	char(128)	Type of host variable
sql_hvar_data	char(8192)	Value of host variable

syssqltrace_info

The **syssqltrace_info** table describes information about the SQL profile trace system.

Column	Type	Description
flags	integer	SQL trace flags
ntraces	integer	Number of items to trace
tracesize	integer	Size of the text to store for each SQL trace item
duration	integer	Trace buffer (in seconds)
sqlseen	int8	Number of SQL items traced since start or resizing
starttime	integer	Time tracing was enabled
memoryused	int8	Number of bytes of memory used by SQL tracing

syssqltrace_iter

The **syssqltrace_iter** table lists the SQL statement iterators.

Column	Type	Description
sql_id	int8	SQL execution ID
sql_address	int8	Address of the SQL statement block
sql_itr_address	int8	Address of the iterator
sql_itr_id	int	Iterator ID
sql_itr_left	int	Iterator ID to the left
sql_itr_right	int	Iterator ID to the right
sql_itr_cost	int	Iterator cost
sql_itr_estrows	int	Iterator estimated rows
sql_itr_numrows	int	Iterator actual rows processed
sql_itr_type	int	Iterator type
sql_itr_misc	int	Iterator miscellaneous flags
sql_it_info	char(256)	Iterator miscellaneous flags displayed as text

syssrcrss

The **syssrcrss** table provides RS secondary server related statistics at the primary server.

Table 111. syssrcrss table column information

Column	Type	Description
address	int8	RS secondary server control block address
server_name	char(128)	Database server name
server_status	char(20)	Quiescent, active, or inactive
connection_status	char(20)	Connected or disconnected
log_transmission_status	char(20)	Active or blocked
next_page_tosend_log_uniq	integer	Unique log ID of next page to send
next_page_tosend_log_page	integer	Page number of next page to send
seq_tosend	integer	Sequence ID of last buffer sent
last_seq_acked	integer	Sequence ID of last buffer acknowledged

syssrcsds

The **syssrcsds** table provides SD secondary server related statistics at the primary server.

The **syssrcsds** table contains the columns that are shown in the following table.

Column	Type	Description
address	int8	SD secondary server control block address
source_server	char(128)	Primary database server name
connection_status	char(20)	Connected or disconnected
last_received_log_uniq	integer	Unique log ID of last log page received
last_received_log_page	integer	Page number of last log page received
next_lpgtoread_log_uniq	integer	Unique log ID of next log page to read
next_lpgtoread_log_page	integer	Page number of next log page to read
last_acked_lsn_uniq	integer	Unique log ID of last LSN acknowledged
last_acked_lsn_pos	integer	Log position of last LSN acknowledged
last_seq_received	integer	Sequence ID of last buffer received
last_seq_acked	integer	Sequence ID of last buffer acknowledged

Column	Type	Description
cur_pagingfile	char(640)	Current® paging file name
cur_pagingfile_size	int8	Current® paging file size
old_pagingfile	char(640)	Old paging file name
old_pagingfile_size	int8	Old paging file size

systabnames

The **systabnames** table describes each table that the database server manages.

Column	Type	Description
partnum	integer	tblspace identifier
dbsname	char(128)	Database name
owner	char(32)	User ID of owner
tabname	char(128)	Table name
collate	char(32)	Collation associated with a database that supports GLS

systhreads

The **systhreads** table provides information about each thread.

Column	Type	Description
th_id	INTEGER	The numeric identifier of the thread.
th_addr	INTEGER	The memory address of the thread control block.
th_joinlist	INTEGER	If a list of the threads are waiting for this thread to exit, the th_joinlist column shows the address of the first thread in the list.
th_joinnext	INTEGER	If a list of the threads are waiting for this thread to exit, the th_joinnext column shows the address of the next thread in the join list.
th_joinee	INTEGER	The address of the thread whose exit this thread is waiting for.
th_name	CHAR(12)	The name of the thread.
th_state	INTEGER	The status code of the thread.
th_priority	INTEGER	The priority of the thread.
th_class	INTEGER	The code for the class of virtual processor that thread will run on.
th_vpid	INTEGER	The ID of the virtual processor that the thread was last scheduled to run on.

Column	Type	Description
th_mtxwait	INTEGER	The address of the mutex that this thread is waiting for.
th_conwait	INTEGER	The address of the condition that this thread is waiting for.
th_waketime	INTEGER	The time of the expiration of the last sleep. The time is calculated by an internal clock. A value of -1 means that the time value is indeterminate.
th_startwait	INTEGER	The time when the last wait began. The time is calculated by an internal clock.
th_startrun	INTEGER	The time when the last execution began. The time is calculated by an internal clock.

systrgrss

The **systrgrss** table provides RS secondary server related statistics at the RS secondary server.

Column	Type	Description
address	int8	RS secondary server control block address
source_server	char(128)	Source server serving the RS secondary server
connection_status	char(20)	Connected or disconnected
last_received_log_uniq	integer	Unique log ID of last log page received
last_received_log_page	integer	Page number of last log page received
last_seq_received	integer	Sequence ID of last buffer received
last_seq_acked	integer	Sequence ID of last buffer acknowledged

systrgsds

The **systrgsds** table provides SD secondary server related statistics at the SD secondary server.

The **systrgsds** table contains these columns:

Column	Type	Description
address	int8	SD secondary server control block address
source_server	char(128)	Source server serving the SD secondary server
connection_status	char(20)	Connected or disconnected
last_received_log_uniq	integer	Unique log ID of last log page received
last_received_log_page	integer	Page number of last log page received

Column	Type	Description
next_lptoread_log_uniq	integer	Unique log ID of next log page to read
next_lptoread_log_page	integer	Page number of next log page to read
last_acked_lsn_uniq	integer	Unique log ID of last LSN acknowledged
last_acked_lsn_pos	integer	Log position of last LSN acknowledged
last_seq_received	integer	Sequence ID of last buffer received
last_seq_acked	integer	Sequence ID of last buffer acknowledged
cur_pagingfile	char(640)	Current® paging file name
cur_pagingfile_size	int8	Current® paging file size
old_pagingfile	char(640)	Old paging file name
old_pagingfile_size	int8	Old paging file size

sysvpprof

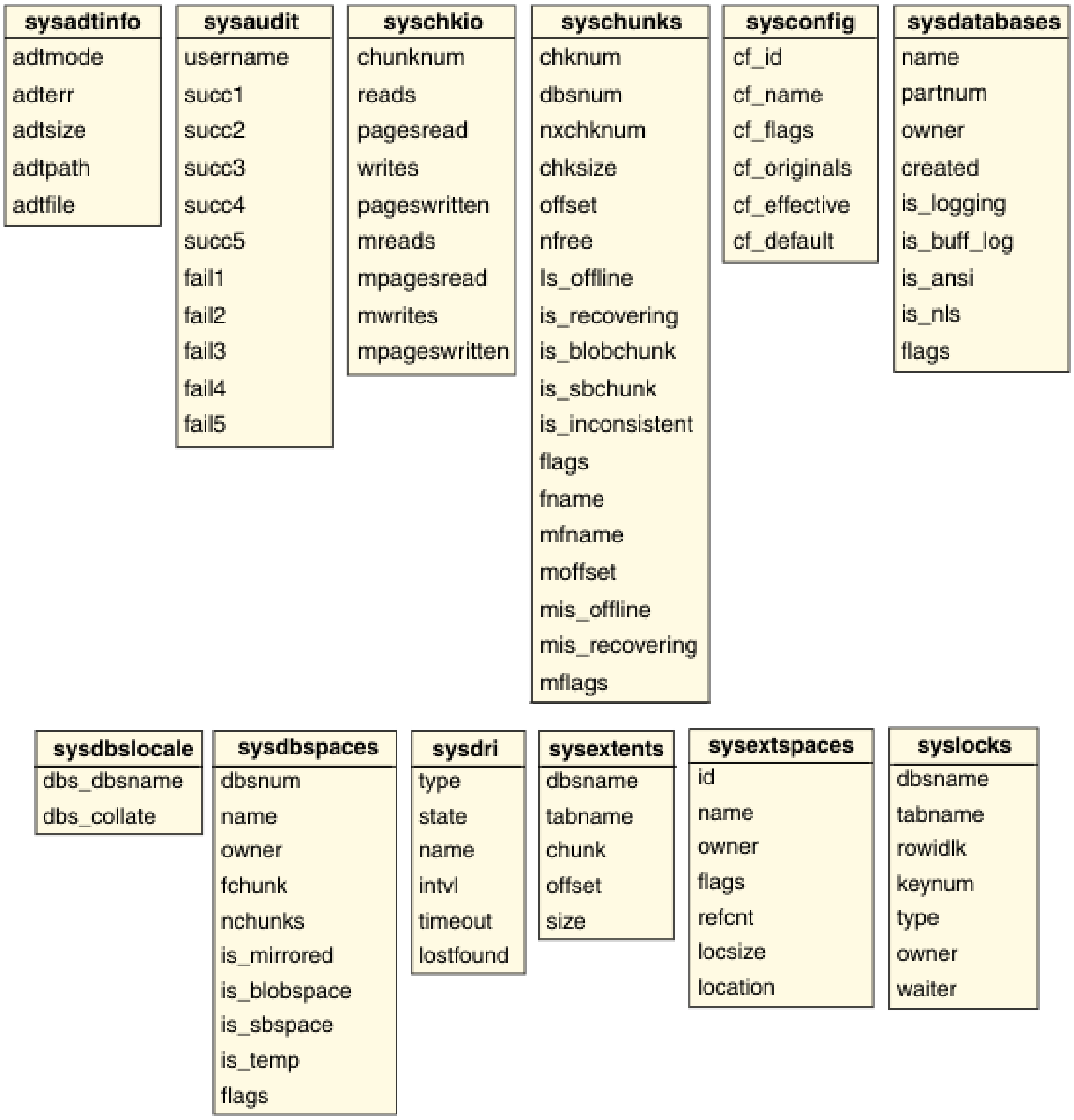
The **sysvpprof** table lists user and system CPU time for each virtual processor.

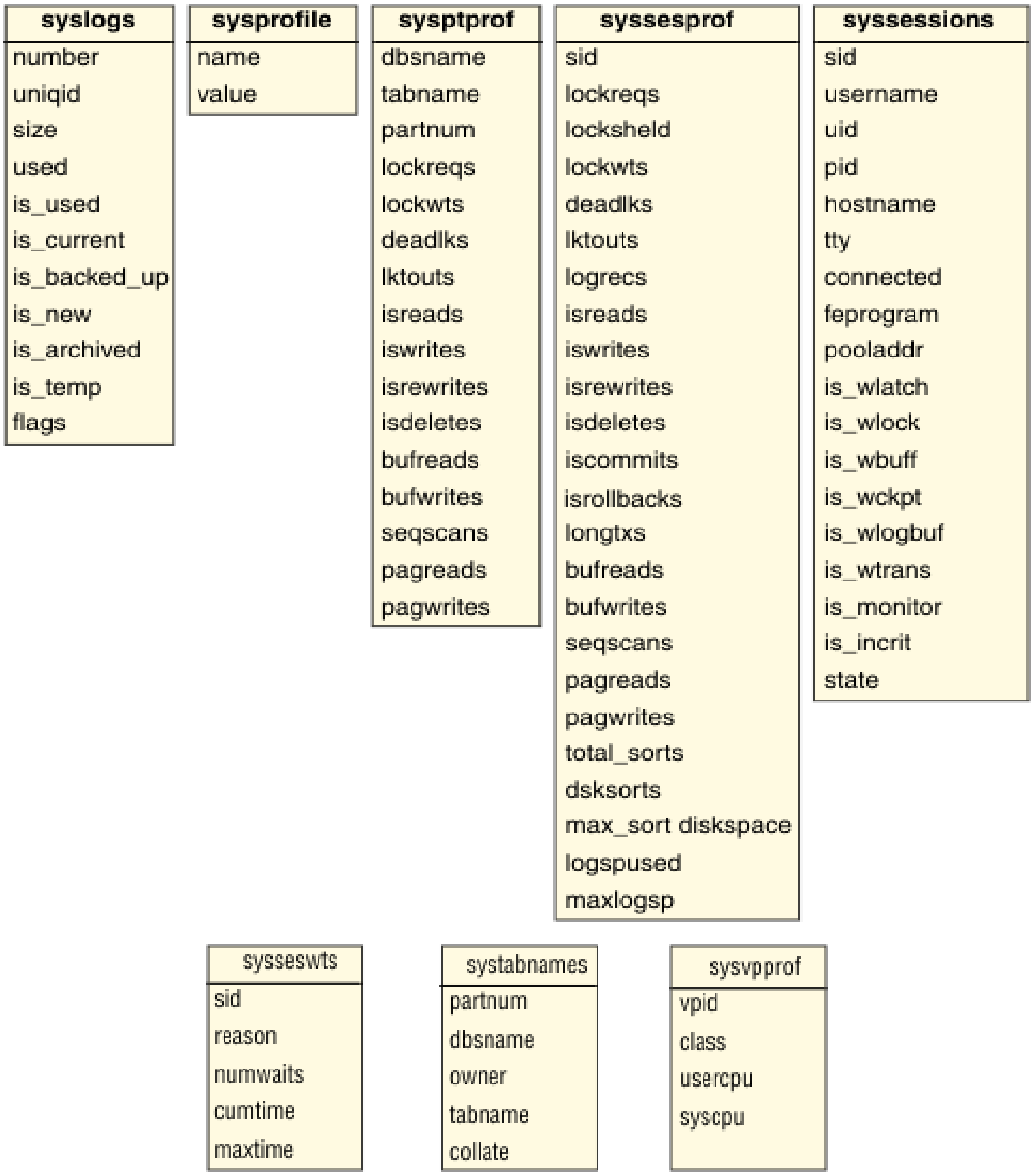
Column	Type	Description
vpid	integer	Virtual processor ID
	char(50)	Type of virtual processor: • cpu • adm • lio • pio • aio • tli • soc • str • shm • opt • msc • adt
usercpu	float	Number of microseconds of user time
syscpu	float	Number of microseconds of system time

The SMI Tables Map

Figure 11: Columns in the SMI tables on page 280 displays the columns in some of the SMI tables.

Figure 11. Columns in the SMI tables





Information from onstat in the SMI Tables

To obtain information provided by the **onstat** utility, you can use SQL to query appropriate SMI tables. The following table indicates which SMI tables to query to obtain the information provided by a given **onstat** option. For descriptions of the **onstat** options, see Monitor the database server status on page 478.

onstat Option	SMI Tables to Query	onstat Fields *Not* in SMI Tables
-d	sysdbspaces syschunks	address bpages
-D	sysdbspaces syschkio	
-F	sysprofile	address flusher snoozer state data

onstat Option	SMI Tables to Query	onstat Fields *Not* in SMI Tables
-g ath	systhreads	
-g dri	sysdri	Last DR CKPT (id/pg)
-g glo	sysvpprof	Listing of virtual processors by
-g ipl	sysipl	
-g rss	sysrsslog systrgrss syssrcrss	
-g his	syssqltracing	
-g sds	syssrcsds systrgsds	
-g smx	syssmx	
-g smx ses	syssmxses	
-k	syslocks	address lklist tblsnum
-l	syslogs sysprofile	All physical-log fields (except numpages and numwrits) All logical-log buffer fields (except numrecs, numpages, and numwrits) address begin % used
-p	sysprofile	
-u	syssessions syssesprof	address wait nreads nwrites

Disk Structures and Storage

In This Chapter

The database server achieves its high performance by managing its own I/O. The database server manages storage, search, and retrieval. As the database server stores data, it creates the structures it needs to search for and retrieve the data later. The database server disk structures also store and track control information needed to manage logging and backups. Database server structures contain all the information needed to ensure data consistency, both physical and logical.

Before you read this chapter, familiarize yourself with the disk-space terms and definitions in the chapter on where data is stored in the *HCL OneDB™ Administrator's Guide*.

This chapter discusses the following topics related to disk data structures:

- Dbspace structure and storage
- Storage of simple large objects
- Sbspace structure
- Time stamps
- Database and table creation: what happens on disk

Dbspace Structure and Storage

This section explores the disk structures and storage techniques that the database server uses to store data in a dbspace.

Structure of the Root Dbspace

As part of disk-space initialization, the database server initializes specific structures in the initial chunk of the root dbspace.

The following structures are initialized:

- Twelve reserved pages
- The first chunk free-list page
- The tblspace tblspace
- The physical log
- The logical-log files
- The database tblspace

The ROOTNAME, ROOTOFFSET, ROOTPATH, and ROOTSIZE configuration parameters specify the size and location of the initial chunk of the root dbspace. If the root dbspace is mirrored, the MIRROROFFSET and MIRRORPATH configuration parameters specify the mirror-chunk location. For more information about these parameters, see Database configuration parameters on page 3.

To see the structure of the root chunk use the oncheck -pe command. For more information, see oncheck -ce, -pe: Check the chunk-free list on page 329.

Reserved Pages

The first 12 pages of the initial chunk of the root dbspace are reserved pages. Each reserved page contains specific control and tracking information used by the database server.

To obtain a listing of the contents of your reserved pages, execute the command oncheck -pr. To also list information about the physical-log and logical-log pages, including the active physical-log pages, run the oncheck -pR command.

The following example shows sample oncheck -pr output for interval checkpoints:

```
 Time of checkpoint      10/25/2005 17:05:20
 Checkpoint Interval    1234
```

The database server also stores current configuration information in a reserved page called PAGE_CONFIG. If you change the configuration parameters from the command line and run the oncheck -pr command without shutting down and restarting the database server, the configuration values in the command output do not match the current values in the reserved pages. The oncheck utility returns a warning message.

The following example shows sample output of the contents of a PAGE_CONFIG reserved page.

```
...

Validating Informix database server reserved pages - PAGE_CONFIG
    ROOTNAME                       rootdbs
```

```
    ROOTPATH                        /home/dyn_srv/root_chunk
    ROOTOFFSET                      4
    ROOTSIZE                        8000
    MIRROR                          0
    MIRRORPATH
    MIRROROFFSET                    0
    PHYSFILE                        1000
    LOGFILES                        5
    LOGSIZE                         500
    MSGPATH                         /home/dyn_srv/online.log
    CONSOLE                             /dev/ttyp5
...                                 ...
```

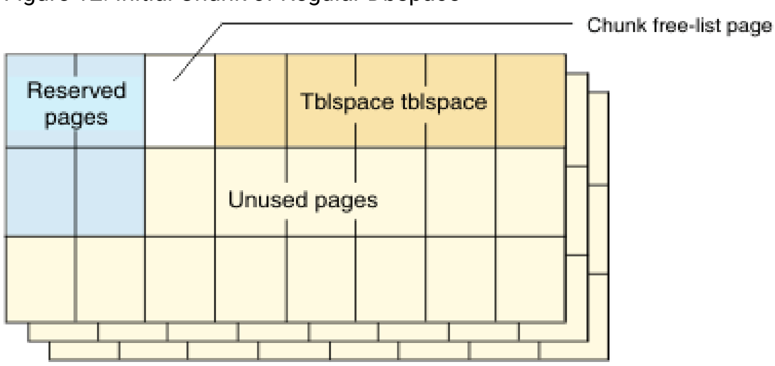
Structure of a Regular Dbspace

After disk-space initialization, you can add new dbspaces. When you create a dbspace, you assign at least one chunk (either raw or cooked disk space) to the dbspace. This chunk is referred to as the initial chunk of the dbspace. Figure 12: Initial Chunk of Regular Dbspace on page 284 illustrates the structure of the initial chunk of a regular (nonroot) dbspace.

When the dbspace is first created, it contains the following structures:

- Two reserved pages
- The first chunk free-list page in the chunk
- The tblspace **tblspace** for this dbspace
- Unused pages

Figure 12. Initial Chunk of Regular Dbspace



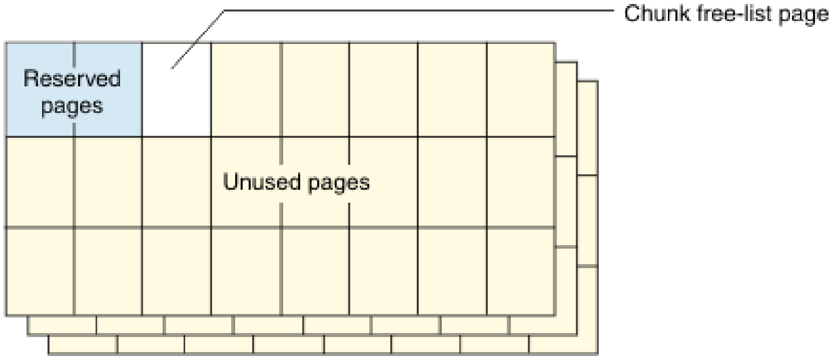
Structure of an Additional Dbspace Chunk

You can create a dbspace that contains more than one chunk. The initial chunk in a dbspace contains the tblspace **tblspace** for the dbspace. Additional chunks do not. When an additional chunk is first created, it contains the following structures:

- Two reserved pages
- The first chunk free-list page
- Unused pages

Figure 13: Additional Dbspace Chunk on page 285 illustrates the structure of all additional chunks in a dbspace. (The structure also applies to additional chunks in the root dbspace.)

Figure 13. Additional Dbspace Chunk



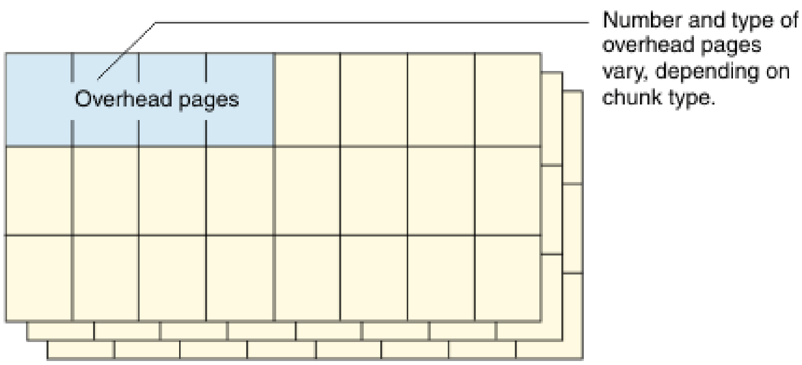
Structure of a Mirror Chunk

Each mirror chunk must be the same size as its primary chunk. When a mirror chunk is created, the database server writes the contents of the primary chunk to the mirror chunk immediately.

The mirror chunk contains the same control structures as the primary chunk. Mirrors of blobspace, sbspace, or dbspace chunks contain the same physical contents as their primary counterpart after the database server brings them online.

Figure 14: Mirror-Chunk Structure on page 285 illustrates the mirror-chunk structure as it appears after the chunk is created.

Figure 14. Mirror-Chunk Structure



The mirror-chunk structure always shows no free space because all of its space is reserved for mirroring. For more information, see the chapter on what is mirroring in the *HCL OneDB™ Administrator's Guide*.

Structure of the Chunk Free-List Page

In every chunk, the page that follows the last reserved page is the first of one or more chunk free-list pages that tracks available space in the chunk. For a non-root chunk, the initial length of the free space is equal to the size of the chunk minus three pages. If an additional chunk free-list page is needed to accommodate new entries, a new chunk free-list page is created in one of the free pages in the chunk. Figure 15: Free-List Page on page 286 illustrates the location of the free-list page.

Use **oncheck -pe** to obtain the physical layout of pages in the chunk. For more information, see oncheck -ce, -pe: Check the chunk-free list on page 329.

Figure 15. Free-List Page

Structure of the Tblspace Tblspace

Each dbspace contains a tblspace called the *tblspace **tblspace*** that describes all tblspaces in the dbspace. When the database server creates a tblspace, it places an entry in the tblspace **tblspace** that describes the characteristics of the newly created tblspace. You cannot drop or move a chunk containing a tblspace **tblspace.**

A dbspace can have a maximum number of 2**20 tblspaces.

The default size of the first and next extents depends on whether the dbspace is the root dbspace or not, as shown in the following table.

Table 112. Default sizes for each extent and type of dbspace

Type of dbspace	Default Size of First Extent	Default Size of Next Extents
Root	• 500 KB for a 2 kilobyte page system • 1000 KB for a 4 kilobyte page system	• 100 KB for a 2 kilobyte page system • 200 KB for a 4 kilobyte page system
Non-root	• 100 KB for a 2 kilobyte page system • 200 KB for a 4 kilobyte page system	• 100 KB for a 2 kilobyte page system • 200 KB for a 4 kilobyte page system

You can specify a non-default size for the first and next extents for a tblspace **tblspace** in the following ways:

- For the root dbspace, set the TBLTBLFIRST and TBLTBLNEXT configuration parameters.
- For non-root dbspaces, use the **onspaces** utility **-ef** and **-en** options when you create a dbspace.

Tblspace tblspace entries

The tblspace **tblspace** describes the characteristics of tblspaces.

To display information on a tblspace, use the oncheck -pt command.

Table 113. tblspace tblspace entries

Component	Description
Page header	24 bytes, standard page-header information

Table 113. tblspace tblspace entries (continued)

Component	Description
Page-ending time stamp	4 bytes
Tblspace header	136 bytes, general tblspace information
Tblspace name	*database.owner.tablename* or *database.owner.indexname* Typically 30-40 bytes long but can be longer, depending on the length of the name.
Column information	8 bytes for each special column A special column is defined as a VARCHAR, BYTE, TEXT, or user-defined data type.
Index information	For attached indexes, each index in the partition has a 20-byte header that contains general information about the index, followed by a 4-byte entry for each column in the index. For detached indexes, a 4-byte entry for each column in the index.
Extent information	A 10-byte entry plus 10 bytes of information for each extent that is allocated to the tblspace. During the defragmentation of the tblspace, more bytes might be used.

Tblspace Numbers

Each tblspace that is described in the tblspace tblspace receives a tblspace number. This tblspace number is the same value that is stored as the **partnum** field in the **systables** system catalog table and as the **partn** field in the **sysfragments** system catalog table.

The following SQL query retrieves the **partnum** for every table in the database (these can be located in several different dbspaces) and displays it with the table name and the hexadecimal representation of **partnum**:

```
SELECT tabname, tabid, partnum, HEX(partnum) hex_tblspace_name FROM systables
```

If the output includes a row with a table name but a **partnum** of 0, this table consists of two or more table fragments, each located in its own tblspace. For example, Figure 16: Output from systables Query with partnum Values on page 288 shows a table called **account** that has **partnum** 0.

Figure 16. Output from systables Query with partnum Values

```
tabname        tabid   partnum          hex_tblspace_name

sysfragments    25     1048611          0x00100023
branch         100     1048612          0x00100024
teller         101     1048613          0x00100025
account        102           0          0x00000000
history        103     1048615          0x00100027
results        104     1048616          0x00100028
```

To obtain the actual tblspace numbers for the fragments that make up the table, you must query the **sysfragments** table for the same database. Figure 17: Output from sysfragments Table with partn Values on page 288 shows that the **account** table from Figure 16: Output from systables Query with partnum Values on page 288 has three table fragments and three index fragments.

Figure 17. Output from sysfragments Table with partn Values

```
tabid  fragtype     partn          hex_tblspace_name

  102  T            1048614        0x00100026
  102  T            2097154        0x00200002
  102  T            3145730        0x00300002
  102  I            1048617        0x00100029
  102  I            2097155        0x00200003
  102  I            3145731        0x00300003
```

Tblspace Number Elements

The first page in a tblspace is logical page 0. (Physical page numbers refer to the address of the page in the chunk.) The root space tblspace **tblspace** is always contained in the first dbspace and on logical page 1 within the tblspace **tblspace**. (The bitmap page is page 0.)

Tblspace Tblspace Size

These tblspace **tblspace** pages are allocated as an extent when the dbspace is initialized. If the database server attempts to create a table, but the tblspace **tblspace** is full, the database server allocates a next extent to the tblspace.

When a table is removed from the dbspace, its corresponding entry in the tblspace **tblspace** is deleted.

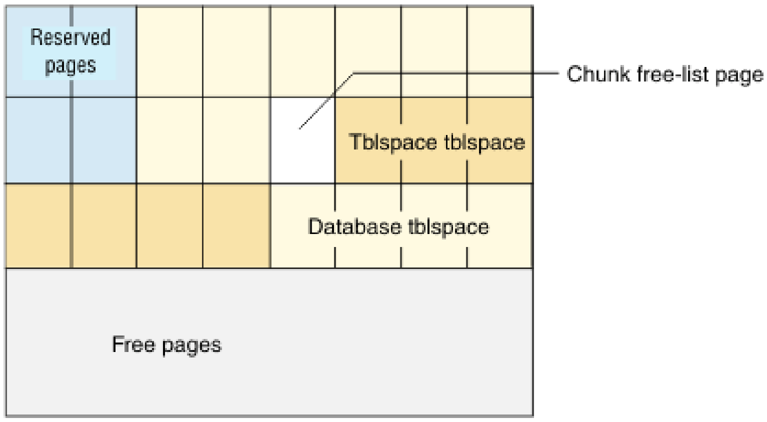
Tblspace Tblspace Bitmap Page

The first page of the tblspace **tblspace**, like the first page of any initial extent, is a bitmap that describes the page fullness of the following pages. Each page that follows has an entry on the bitmap page. If needed, additional bitmap pages are located throughout the contiguous space allocated for the tblspace, arranged so that each bitmap describes only the pages that follow it, until the next bitmap or the end of the dbspace. Bitmap pages fall at distinct intervals within tblspaces pages. Each bitmap page describes a fixed number of pages that follow it.

Structure of the Database Tblspace

The database tblspace appears only in the initial chunk of the root dbspace. The database tblspace contains one entry for each database managed by the database server. Figure 18: Database Tblspace Location in Initial Chunk of Root Dbspace on page 289 illustrates the location of the database tblspace.

Figure 18. Database Tblspace Location in Initial Chunk of Root Dbspace



Database Tblspace Number

The tblspace number of the database tblspace is always 0x100002. This tblspace number appears in an **onstat -t** listing if the database tblspace is active.

Database Tblspace Entries

Each database tblspace entry includes the following five components:

- Database name
- Database owner
- Date and time that the database was created
- The tblspace number of the **systables** system catalog table for this database
- Flags that indicate logging mode

The database tblspace includes a unique index on the database name to ensure that every database is uniquely named. For any database, the **systables** table describes each permanent table in the database. Therefore, the database tblspace only points to the detailed database information located elsewhere.

When the root dbspace is initialized, the database tblspace first extent is allocated. The initial-extent size and the next-extent size for the database tblspace are four pages. You cannot modify these values.

Structure and Allocation of an Extent

This section covers the following topics:

- Extent structure
- Next-extent allocation

Extent Structure

An extent is a collection of contiguous pages within a dbspace. Every permanent database table has two extent sizes associated with it. The initial-extent size is the number of kilobytes allocated to the table when it is first created. The next-extent size is the number of kilobytes allocated to the table when the initial extent, and every extent thereafter, becomes full.

Blobspaces do not use extents.

For specific instructions on how to specify and calculate the size of an extent, see your *HCL OneDB™ Performance Guide*.

Extent size

The default size for first and next extents is 16 kilobytes. If this transforms to fewer than 4 pages in a particular dbspace, the database server uses the minimum extent size of 4 pages. If a dbspace has a size of 8 kilobytes, which transforms to 2 pages, the database server increases the extent size to 32 kilobytes.

The maximum size of an extent is 2**31 pages, equivalent to the maximum chunk size.

If the chunk is smaller than the maximum size, the maximum extent size depends on the contiguous space available in the chunk.

Tblspaces that hold *index fragments* follow different rules for extent size. The database server bases the extent size for these tblspaces on the extent size for the corresponding table fragment. The database server uses the ratio of the row size to index key size to assign an appropriate extent size for the index tblspace (see the sections on estimating index page size and fragmenting table indexes in the *HCL OneDB™ Performance Guide*).

The maximum number of extents for a partition is 32767.

Page Types Within a Table Extent

Within the extent, individual pages contain different types of data. Extent pages for a table can be separated into the following categories:

- Data pages

  Data pages contain the data rows for the table.
- Bitmap pages

  Bitmap pages contain control information that monitors the fullness of every page in the extent.
- Blobpages

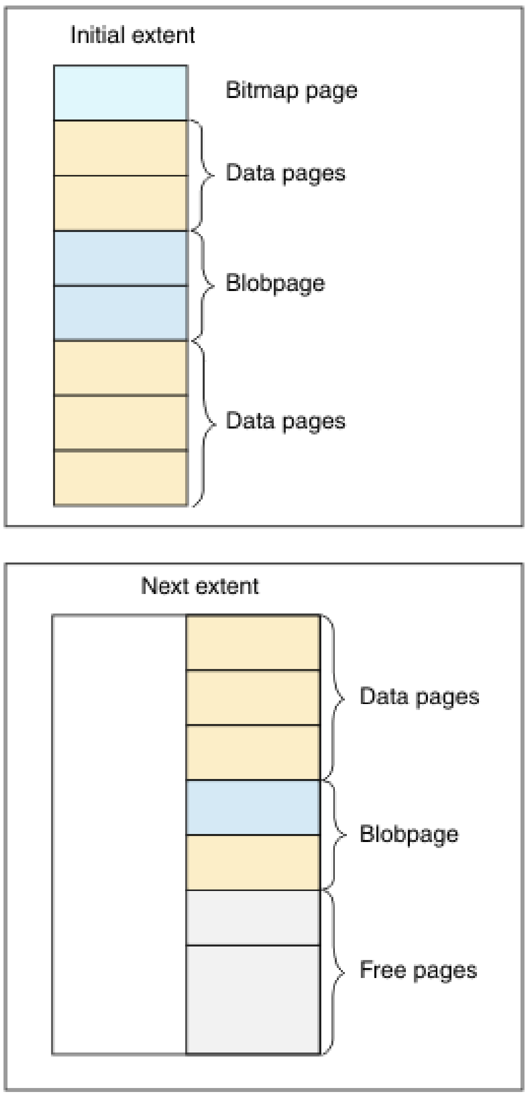
  Blobpages contain TEXT and BYTE data that is stored with the data rows in the dbspace. TEXT and BYTE data that resides in a blobspace is stored in blobpages, a structure that is completely different than the structure of a dbspace blobpage.

- Free pages

  Free pages are pages in the extent that are allocated for tblspace use, but whose function has not yet been defined. Free pages can be used to store any kind of information: data, including TEXT or BYTE data types; index; or bitmap.

Figure 19: Extent Structure of a Table on page 291 illustrates the possible structure of a nonfragmented table with an initial-extent size of 8 pages and a next-extent size of 16 pages.

Figure 19. Extent Structure of a Table



Page Types Within an Index Extent

The database server stores index pages into different tblspaces than the table with which it is associated. Within the extent, individual index pages contain different types of data. Index pages can be separated into the following categories:

- Index pages (root, branch, and leaf pages)

  Index pages contain the index information for the table.

- Bitmap pages

  Bitmap pages contain control information that monitors the fullness of every page in the extent.

- Free pages

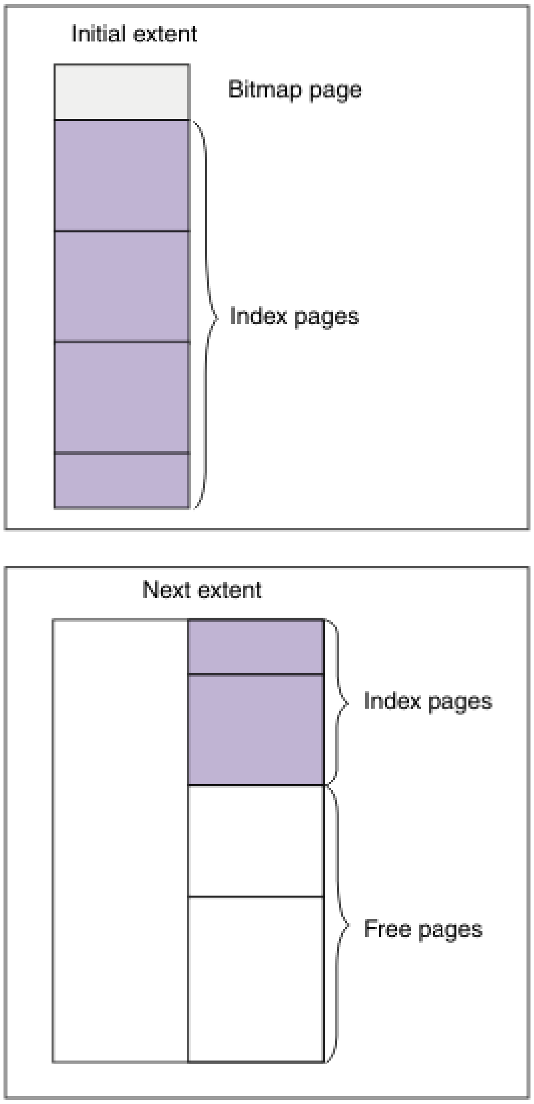
  Free pages are pages in the extent that are allocated for tblspace use, but whose function has not yet been defined. Free pages can be used to store any kind of information: data, index, TEXT or BYTE data, or bitmap.

All indexes are detached unless you explicitly specify attached indexes.

Important: An extent that is allocated for a table fragment does not contain index pages. Index pages for a fragmented table always reside in a separate tblspace. For more information, see fragmenting table indexes in the chapter on table fragmentation and PDQ in the *HCL OneDB™ Administrator's Guide*.

Figure 20: Extent Structure of an Index on page 293 illustrates the extent structure of an index.

Figure 20. Extent Structure of an Index



Next-Extent Allocation

After the initial extent fills, the database server attempts to allocate another extent of contiguous disk space. The procedure that the database server follows is referred to as next-extent allocation.

Extents for a tblspace are tracked as one component of the tblspace **tblspace** information for the table. The maximum number of extents allocated for any tblspace is application and machine dependent because it varies with the amount of space available on the tblspace **tblspace** entry.

Next-Extent Size

The number of kilobytes that the database server allocates for a next extent is, in general, equal to the size of a next extent, as specified in the SQL statement CREATE TABLE. However, the actual size of the next-extent allocation might deviate from the specified size because the allocation procedure takes into account the following three factors:

- Number of existing extents for this tblspace
- Availability of contiguous space in the chunk and dbspace
- Location of existing tblspace extents

The effect of each of these factors on next-extent allocation is explained in the paragraphs that follow and in Figure 21: Next-Extent Allocation Strategies on page 295.

Extent size doubling

For permanent tables or user-defined temporary tables, the size of the next extent for every allocation is automatically doubled. The size doubles up to 128 kilobytes (KB). For example, if you create a table with the NEXT SIZE equal to 15 KB, the database server allocates the first extent at a size of 15 KB. The next extent is allocated at 30 KB, and the extent after that is allocated at 60 KB. When the extent size reaches 128 KB, the size is doubled only when the remaining space in the table is less than 10% of the total allocated space in the table.

For system-created temporary tables, the next-extent size begins to double after 4 extents have been added.

Lack of Contiguous Space

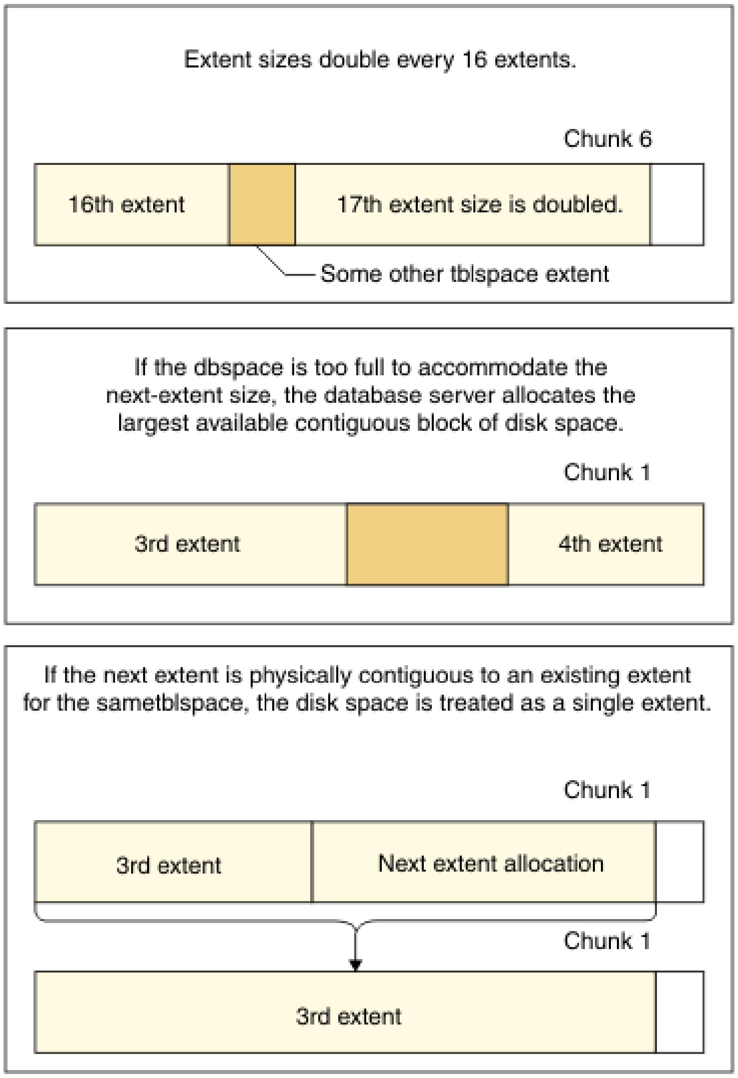
If the database server cannot find available contiguous space in the first chunk equal to the size specified for the next extent, it extends the search to the next chunk in the dbspace. Extents are not allowed to span chunks.

If the database server cannot find adequate contiguous space anywhere in the dbspace, it allocates to the table the largest available amount of contiguous space. (The minimum allocation is four pages. The default value is eight pages.) No error message is returned if an allocation is possible, even when the amount of space allocated is less than the requested amount.

Merge of Extents for the Same Table

If the disk space allocated for a next extent is physically contiguous with disk space already allocated to the same table, the database server allocates the disk space but does not consider the new allocation as a separate extent. Instead, the database server extends the size of the existing contiguous extent. Thereafter, all disk-space reports reflect the allocation as an extension of the existing extent. That is, the number of extents reported is always the number of physically distinct extents, not the number of times a next extent has been allocated plus one (the initial extent). Figure 21: Next-Extent Allocation Strategies on page 295 illustrates extent-allocation strategies.

Figure 21. Next-Extent Allocation Strategies



After disk space is allocated to a tblspace as part of an extent, the space remains dedicated to that tblspace even if the data contained in it is deleted. For alternative methods of reclaiming this empty disk space, see your *HCL OneDB™ Performance Guide*.

Structure and Storage of a Dbspace Page

The basic unit of database server I/O is a page. Page size might vary among computers.

In HCL OneDB™, the page size depends on the operating system.

Rows in Nonfragmented Tables

The database server can store rows that are longer than a page. The database server also supports the VARCHAR data type, which results in rows of varying length. As a result, rows do not conform to a single format.

Rows within a table are not necessarily the same length if the table contains one or more columns of type VARCHAR. In addition, the length of a row in such a table might change when an end user modifies data contained in the VARCHAR column.

The length of a row can be greater than a page.

TEXT and BYTE data is not stored within the data row. Instead, the data row contains a 56-byte descriptor that points to the location of the data. The descriptor can point to a dbspace page.

The descriptor can point to a blobspace blobpage.

The descriptor can point to a blobspace blobpage. If you are using the Optical Subsystem, the descriptor can also point to an optical-storage subsystem.

For instructions about how to estimate the length of fixed-length and variable-length data rows, see your *HCL OneDB™ Performance Guide*.

Definition of Rowid

HCL OneDB™ uses two different types of rowids to identify data in tables:

- *Serial rowid*

  These rowids are fields in a table and are assigned to tables created with the WITH ROWID option.

- *Internal rowid*

  The database server identifies each data row in a table with a unique internal rowid. This rowid identifies the location of the row within the dbspace.

  To obtain the internal rowids for a table, use the **oncheck -pD** option. For more information, see oncheck -cd and oncheck -cD commands: Check pages on page 327.

In a nonfragmented table, the term *rowid* refers to a unique 4-byte integer that defines the physical location of the row in the table. The page that contains the first byte of the data row is the page that is specified by the rowid. This page is called the data row *home page*.

Fragmented tables can also have rowids, but they are implemented in a different way. For more information on this topic, see Rows in Fragmented Tables on page 297.

Use of Rowids

Every data row in a nonfragmented table is uniquely identified by an unchanging rowid. When you create an index for a nonfragmented table, the rowid is stored in the index pages associated with the table to which the data row belongs. When the database server requires a data row, it searches the index to find the key value and uses the corresponding rowid to locate the requested row. If the table is not indexed, the database server might sequentially read all the rows in the table.

Eventually, a row might outgrow its original storage location. If this occurs, a *forward pointer* to the new location of the data row is left at the position defined by the rowid. The forward pointer is itself a rowid that defines the page and the location on the page where the data row is now stored.

Rows in Fragmented Tables

Unlike rows in a nonfragmented table, the database server does *not* assign a rowid to rows in fragmented tables. If you want to access data by rowid, you must explicitly create a rowid column as described in your *HCL OneDB™ Performance Guide*. If user applications attempt to reference a rowid in a fragmented table that does not contain a rowid that you explicitly created, the database server returns an appropriate error code to the application.

Access to Data in Fragmented Tables with Rowid

From the viewpoint of an application, the functionality of a rowid column in a fragmented table is identical to the rowid of a nonfragmented table. However, unlike the rowid of a nonfragmented table, the database server uses an index to map the rowid to a physical location.

When the database server accesses a row in a fragmented table using the rowid column, it uses this index to look up the physical address of the row before it attempts to access the row. For a nonfragmented table, the database server uses direct physical access without an index lookup. As a consequence, accessing a row in a fragmented table using rowid takes slightly longer than accessing a row using rowid in a nonfragmented table. You should also expect a small performance impact on the processing of inserts and deletes due to the cost of maintaining the rowid index for fragmented tables.

Primary-key access can lead to significantly improved performance in many situations, particularly when access is in parallel.

Recommendations on Use of Rowid

It is recommended that application developers use primary keys as a method of access rather than rowids. Because primary keys are defined in the ANSI specification of SQL, using them to access data makes your applications more portable.

For a complete description on how to define and use primary keys to access data, see the *HCL OneDB™ Guide to SQL: Reference* and the *HCL OneDB™ Guide to SQL: Tutorial*.

Data-Row Format and Storage

The variable length of a data row has the following consequences for row storage:

- A page might contain one or more whole rows.
- A page might contain portions of one or more rows.
- A page might contain a combination of whole rows and partial rows.
- An updated row might increase in size and become too long to return to its original storage location in a row.

The following paragraphs describe the guidelines that the database server follows during data storage.

Storage of Row

To minimize retrieval time, rows are not broken across page boundaries unnecessarily. Rows that are shorter than a page are always stored as whole rows. A page is considered *full* when the count of free bytes is less than the number of bytes needed to store a row of maximum size.

Location of Rows

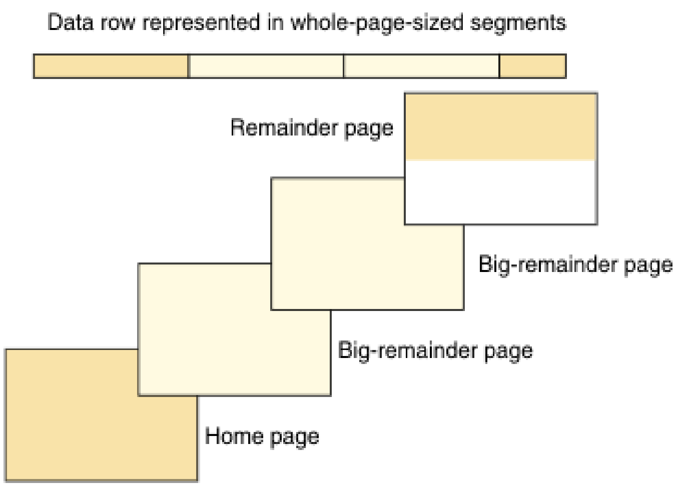
When the database server receives a row that is longer than a page, the row is stored in as many whole pages as required. The database server then stores the trailing portion in less than a full page.

The page that contains the first byte of the row is the row home page. The number of the home page becomes the logical page number contained in the rowid. Each full page that follows the home page is referred to as a big-remainder page. If the trailing portion of the row is less than a full page, it is stored on a remainder page.

After the database server creates a remainder page to accommodate a long row, it can use the remaining space in this page to store other rows.

Figure 22: Remainder Pages on page 298 illustrates the concepts of home page, big-remainder page, and remainder page.

Figure 22. Remainder Pages



Page Compression

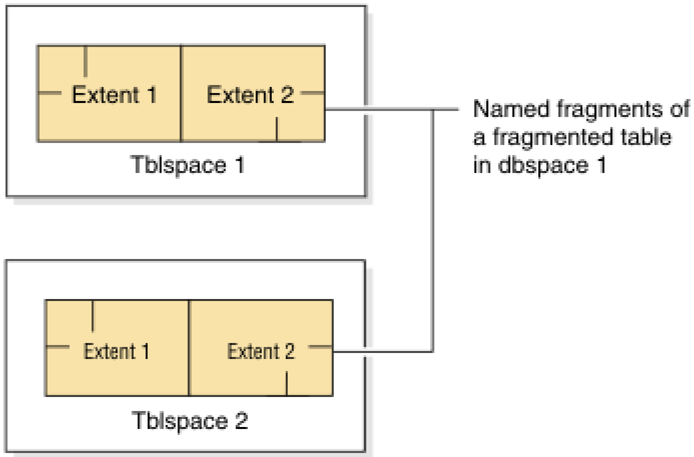
Over time, the free space on a page can become fragmented. When the database server attempts to store data, it first checks row length against the number of free bytes on a page to determine if the row fits. If adequate space is available, the database server checks if the page contains adequate contiguous free space to hold the row (or row portion). If the free space is not contiguous, the database server calls for page compression.

Structure of Fragmented Tables

Although table fragmentation is transparent to applications, as database server administrator you should be aware of how the database server allocates disk space for table fragments and how the database server identifies rows in those fragments.

Each table fragment has its own tblspace with a unique *tblspace_id* or *fragment_id*. Figure 23: Disk Structures for a Fragmented Table on page 299 shows the disk allocation for a fragmented table that resides in named fragments of the same dbspace.

Figure 23. Disk Structures for a Fragmented Table



Attached Indexes

With an attached index, the index and data are fragmented in the same way. You can decide whether to store the index pages with the corresponding data pages in the same dbspace or store them in separate dbspaces. For information on choosing a fragmentation strategy, see the *HCL OneDB™ Performance Guide*.

Detached Indexes

For detached indexes, the table fragment and index fragment are stored in tblspaces in separate dbspaces.

Structure of B-Tree Index Pages

This section provides general information about the structure of B-tree index pages. It is designed as an overview for the interested reader. For more information on B-tree indexes, see your *HCL OneDB™ Performance Guide*.

Definition of B-tree terms

The database server uses a B-tree structure to organize index information.

Figure 24: Full B-Tree Structure on page 300 shows that a fully developed B-tree index is composed of the following three different types of index pages or nodes:

- One *root node*

  A root node contains node pointers to branch nodes.

- Two or more *branch nodes*

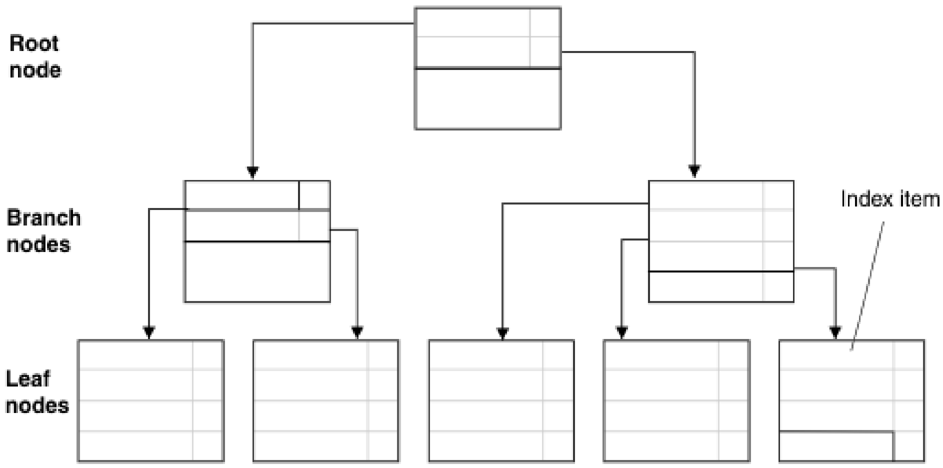
  A branch node contains pointers to leaf nodes or other branch nodes.

- Many *leaf nodes*

  A leaf node contains index items and horizontal pointers to other leaf nodes.

Each node serves a different function. The following sections describe each node and the role that it plays in indexing.

Figure 24. Full B-Tree Structure



Index items

The fundamental unit of an index is the *index item*. An index item contains a key value that represents the value of the indexed column for a particular row. An index item also contains rowid information that the database server uses to locate the row in a data page.

Index nodes

A node is an index page that stores a group of index items.

Forest of trees indexes as alternatives to traditional B-Tree indexes

Unlike a traditional B-tree index, a forest of trees index is a large B-tree index that is divided into smaller subtrees with multiple root nodes and fewer levels. You can create a forest of trees index as an alternative to a B-tree index when you want to alleviate root node contention and allow more concurrent users to access the index without waiting.

Logical Storage of Indexes

This section presents an overview of how the database server creates and fills an index.

Creation of Root and Leaf Nodes

When you create an index for an empty table, the database server allocates a single index page. This page represents the root node and remains empty until you insert data in the table.

At first, the root node functions in the same way as a leaf node. For each row that you insert into the table, the database server creates and inserts an index item in the root node. Figure 25: Root Node on page 301 illustrates how a root node appears before it fills.

Figure 25. Root Node

Root node 1

Albertson	rowid information
Baxter	rowid information
Beatty	rowid information
Currie	rowid information
Keyes	rowid information
Lawson	rowid information
Mueller	rowid information

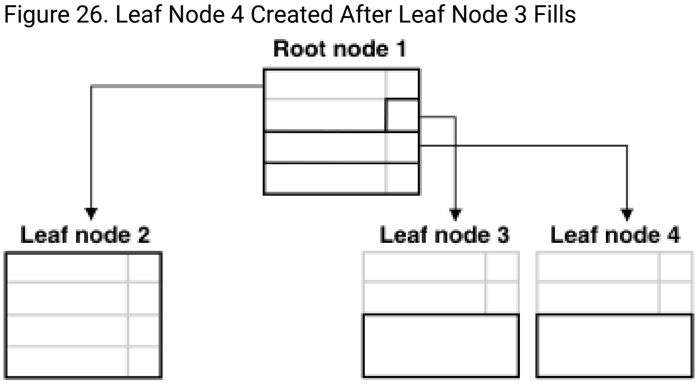
When the root node becomes full of index items, the database server splits the root node by performing the following steps:

- Creates two leaf nodes
- Moves approximately half of the root-node entries to each of the newly created leaf nodes
- Puts pointers to leaf nodes in the root node

As you add new rows to a table, the database server adds index items to the leaf nodes. When a leaf node fills, the database server creates a new leaf node, moves part of the contents of the full index node to the new node, and adds a node pointer to the new leaf node in the root node.

For example, suppose that leaf node 3 in Figure 26: Leaf Node 4 Created After Leaf Node 3 Fills on page 301 becomes full. When this situation occurs, the database server adds yet another leaf node. The database server moves part of the records from leaf node 3 to the new leaf node, as Figure 26: Leaf Node 4 Created After Leaf Node 3 Fills on page 301 shows.

Figure 26. Leaf Node 4 Created After Leaf Node 3 Fills



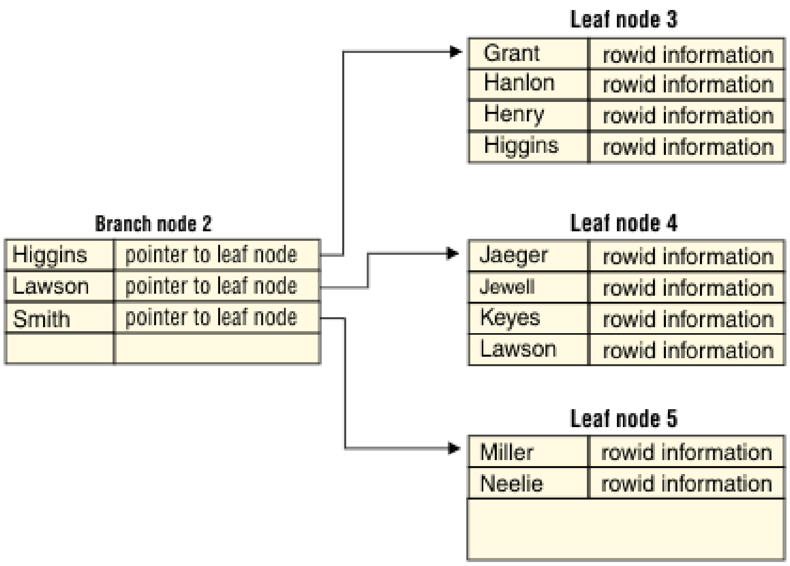
Creation of branch nodes

Eventually, as you add rows to the table, the database server fills the root node with node pointers to all the existing leaf nodes. When the database server splits yet another leaf node, and the root node has no room for an additional node pointer, the following process occurs.

The database server splits the root node and divides its contents among two newly created branch nodes. As index items are added, more and more leaf nodes are split, causing the database server to add more branch nodes. Eventually, the root node fills with pointers to these branch nodes. When this situation occurs, the database server splits the root node again. The database server then creates yet another branch level between the root node and the lower branch level. This process results in a four-level tree, with one root node, two branch levels, and one leaf level. The B-tree structure can continue to grow in this way to a maximum of 20 levels.

Branch nodes can point either to other branch nodes below them (for large indexes of four levels or more) or to leaf nodes. In Figure 27: Typical Contents of a Branch Node on page 302, the branch node points to leaf nodes only. The first item in the left branch node contains the same key value as the largest item in the leftmost leaf node and a node pointer to it. The second item contains the largest item in the next leaf node and a node pointer to it. The third item in the branch node contains only a pointer to the next higher leaf node. Depending on the index growth, this third item can contain the actual key value in addition to the pointer at a later point during the life span of the index.

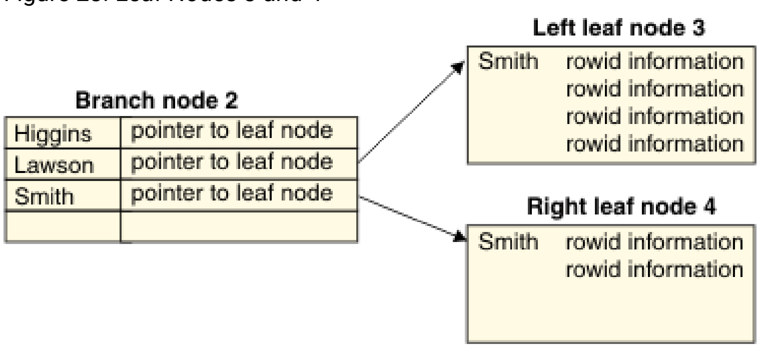
Figure 27. Typical Contents of a Branch Node



Duplicate Key Values

Duplicate key values occur when the value of an indexed column is identical for multiple rows. For example, suppose that the third and fourth leaf nodes of a B-tree structure contain the key value `Smith`. Suppose further that this value is duplicated six times, as Figure 28: Leaf Nodes 3 and 4 on page 302 illustrates.

Figure 28. Leaf Nodes 3 and 4



The first item on the third leaf page contains the duplicate key value, `Smith`, and the rowid information for the first physical row in the table that contains the duplicate key value. To conserve space, the second item does not repeat the key value `Smith` but instead contains just the rowid information. This process continues throughout the page; no other key values are on the leaf, only rowid information.

The first item on the fourth leaf page again contains the duplicated key value and rowid information. Subsequent items contain only rowid information.

Now consider the branch node. The third item in the branch node contains the same key value and rowid as the largest item in the third leaf node and a node pointer to it. The fourth item would contain only a node pointer to the fourth leaf node, thus saving the space of an additional duplicate key value.

Key-Value Locking

To increase concurrency, the database server supports *key-value* locking in the B-tree index. Key-value locking locks only the value of the key instead of the physical location in the B-tree index.

One of the most important uses for key-value locking is to assure that a unique key remains unique through the end of the transaction that deleted it. Without this protection mechanism, user A might delete a unique key within a transaction, and user B might insert a row with the same key before the transaction commits. This scenario makes rollback by user A impossible. Key-value locking prevents user B from inserting the row until the end of user A's transaction.

Adjacent Key Locking

With Repeatable Read isolation level, the database server is required to protect the *read set*. The read set consists of the rows that meet the filters in the WHERE clause of the query. To guarantee that the rows do not change, the database server obtains a lock on the index item that is adjacent to the right-most item of the read set.

Freed Index Pages

When the database server physically removes an index item from a node and frees an index page, the freed page is reused.

Filling Indexes

When you create an index, you can specify how densely or sparsely filled you want the index. The index fill factor is a percentage of each index page that will be filled during the index build. Use the FILLFACTOR option of the CREATE INDEX statement or the FILLFACTOR configuration parameter to set the fill factor. This option is particularly useful for indexes that you do not expect to grow after they are built. For additional information about the FILLFACTOR option of the CREATE INDEX statement, see the *HCL OneDB™ Guide to SQL: Syntax*.

Calculating the Length of Index Items

For data types other than VARCHAR, the length of an index item is calculated by adding the length of the key value plus 5 bytes for each rowid information associated with the key value.

The key values in an index are typically of fixed length. If an index holds the value of one or more columns of the VARCHAR data type, the length of the key value is at least the sum of the length-plus-one of each VARCHAR value in the key.

In HCL OneDB™, the maximum length of a key value is 390 bytes. The combined size of VARCHAR columns that make up a key must be less than 390, minus an additional byte for each VARCHAR column. For example, the key length of the index that the database server builds for the following statements equals 390, or `((255+1) + (133+1))`:

```
CREATE TABLE T1 (c1 varchar(255, 10), c2 varchar(133, 10));
CREATE INDEX I1 on T1(c1, c2);
```

Functional Indexes

A *functional index* is one in which all keys derive from the results of a function. If you have a column of pictures, for example, and a function to identify the predominant color, you can create an index on the result of the function. Such an index would enable you to quickly retrieve all pictures having the same predominant color, without re-executing the function.

A functional index uses the same B-tree structure as any other B-tree index. The only difference is that the determining function is applied during an insert or an update whenever the column that is the argument to the function changes. For more information on the nature of functional indexes, refer to your *HCL OneDB™ Performance Guide*.

To create a functional index, use the CREATE FUNCTION and CREATE INDEX statements. For more information on these statements, refer to the *HCL OneDB™ Guide to SQL: Syntax*.

Structure of R-Tree Index Pages

An index structure that relies on one-dimensional ordering of key values does not work for spatial data; for example, two dimensional geometric shapes such as circles, squares, and triangles. Efficient retrieval of spatial data, such as the data used in geographic information systems (GIS) and computer-aided design (CAD) applications, requires an access method that handles multidimensional data. The database server implements an R-tree index to access spatial data efficiently. For information about the structure of index pages, refer to the *R-Tree Index User's Guide*.

Storage of Simple Large Objects

This section explains the structures and storage techniques that the database server uses to store simple large objects (TEXT or BYTE data).

Structure of a Blobspace

When you create a blobspace, you can specify the effective size of the data pages, which are called blobpages. The blobpage size for the blobspace is specified when the blobspace is created. Blobpage size must be a multiple of page size. (For information on determining database server page size, see the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.) All blobpages within a blobspace are the same size, but the size of the blobpage can vary between blobspaces. Blobpage size can be greater than the page size because data stored in a blobspace is never written to the page-sized buffers in shared memory.

The advantage of customizing the blobpage size is storage efficiency. Within a blobspace, TEXT and BYTE data is stored in one or more blobpages, but simple large objects do not share blobpages. Storage is most efficient when the TEXT or BYTE data is equal to or slightly smaller than the blobpage size.

The blobspace free-map pages and bitmap pages are the size specified as a database server page, which enables them to be read into shared memory and to be logged.

When the blobspace is first created, it contains the following structures:

- Blobspace free-map pages
- The blobspace bitmap that tracks the free-map pages
- Unused blobpages

Structure of a Dbspace Blobpage

TEXT or BYTE data that is stored in the dbspace is stored in a blobpage. The structure of a dbspace blobpage is similar to the structure of a dbspace data page. The only difference is an extra 12 bytes that can be stored along with the TEXT or BYTE data in the data area.

Simple large objects can share dbspace blobpages if more than one simple large object can fit on a single page, or if more than one trailing portion of a simple large object can fit on a single page.

For a discussion of how to estimate the number of dbspace blobpages needed for a specific table, see your *HCL OneDB™ Performance Guide*.

Each segment of TEXT or BYTE data stored in a dbspace page might be preceded by up to 12 bytes of information that does not appear on any other dbspace page. These extra bytes are overhead.

Simple-Large-Object Storage and the Descriptor

Data rows that include TEXT or BYTE data do not include the data in the row itself. Instead, the data row contains a 56-byte descriptor with a forward pointer (rowid) to the location where the first segment of data is stored.

The descriptor can point to one of the following items:

- A page (if the data is stored in a dbspace)
- A blobpage (if the data is stored in a blobspace)
- An optical platter (if you are using the Optical Subsystem)

Creation of Simple Large Objects

When a row that contains TEXT or BYTE data is to be inserted, the simple large objects are created first. After the simple large objects are written to disk, the row is updated with the descriptor and inserted.

When a row that contains TEXT or BYTE data is to be inserted, the simple large objects are created first. After the simple large objects are written to disk (or optical medium), the row is updated with the descriptor and inserted.

Deletion or Insertion of Simple Large Objects

The database server cannot modify simple large objects. It can only insert or delete them. Deleting a simple large object means that the database server frees the space consumed by the deleted object for reuse.

When TEXT or BYTE data is updated, a new simple large object is created, and the data row is updated with the new blob descriptor. The old image of the row contains the descriptor that points to the obsolete value for the simple large object. The space consumed by the obsolete simple large object is freed for reuse after the update is committed. Simple large objects are automatically deleted if the rows that contain their blob descriptors are deleted. (Blobpages that stored a deleted simple large object are not available for reuse until the logical log that contains the original INSERT record for the deleted simple large object is backed up. For more information, see backing up logical-log files to free blobpages in the chapter on what is the logical log in the *HCL OneDB™ Administrator's Guide*.)

Size Limits for Simple Large Objects

The largest simple large object that the blob descriptor can accommodate is (2^{31} - 1), or about 2 gigabytes.

Blobspace Page Types

Every blobspace chunk contains three types of pages:

- A blobspace free-map page
- A bitmap page
- Blobpages

Blobspace Free-Map Page

The blobspace free-map page identifies unused blobpages so that the database server can allocate them as part of simple-large-object creation. When a blobpage is allocated, the free-map entry for that page is updated. All entries for a single simple large object are linked.

A blobspace free-map page is the size of one database server page. Each entry on a free-map page is 8 bytes, stored as two 32-bit words, as follows:

- The first bit in the first word specifies whether the blobpage is free or used.
- The next 31 bits in the first word identify the logical-log file that was current when this blobpage was written. (This information is needed for logging TEXT or BYTE data.)
- The second word contains the tblspace number associated with the simple large object stored on this page.

The number of entries that can fit on a free-map page depends on the page size of your computer. The number of free-map pages in a blobspace chunk depends on the number of blobpages in the chunk.

Blobspace Bitmap Page

The blobspace bitmap page tracks the fullness and number of blobspace free-map pages in the chunk. Each blobspace bitmap page is capable of tracking a quantity of free-map pages. The size of the blobspace bitmap page depends on the size of the system page. If the system page is 2K, the blobspace bitmap page can track 2,032,128 blobpages. If the system page is 4K, the blobspace bitmap page can track 8,258,048 blobpages.

Blobpage

The blobpage contains the TEXT or BYTE data. Blobpage size is specified by the database server administrator who creates the blobspace. Blobpage size is specified as a multiple of the page size.

Structure of a Blobspace Blobpage

The storage strategy used to store simple large objects in a blobspace differs from the dbspace storage strategy. The database server does not combine whole simple large objects or portions of a simple large object on a single blobspace blobpage. For example, if blobspace blobpages are 24 kilobytes each, a simple large object that is 26 kilobytes is stored on two 24-kilobyte pages. The extra 22 kilobytes of space remains unused.

The structure of a blobpage includes a blobpage header, the TEXT or BYTE data, and a page-ending time stamp. The blobpage header includes, among other information, the page-header time stamp and the blob time stamp associated with the forward pointer in the data row. If a simple large object is stored on more than one blobpage, a forward pointer to the next blobpage and another blob time stamp are also included in the blobpage header.

Sbspace Structure

An sbspace is similar to a blobspace except that it holds smart large objects.

When an sbspace is created in a database, it contains an sbspace descriptor. Each sbspace chunk contains the following structures:

- Sbspace chunk descriptors
- Chunk free-page list
- An sbspace metadata area (up to one for each chunk)
- Reserved data areas (up to two for each chunk)
- User-data areas (up to two for each chunk)

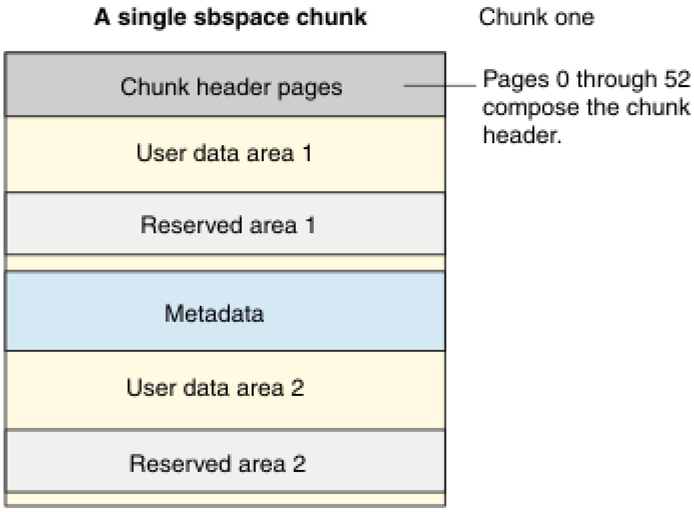
For best performance, it is recommended that the metadata area be located in the middle of the sbspace. The database server automatically places the metadata area in the correct location. However, to specify the location of the metadata area, specify the **-Mo** flag in the **onspaces** command.

If you do not specify the size of the metadata area in the **-Ms** flag of the **onspaces** command, the database server uses the value of AVG_LO_SIZE (defaults to 8 kilobytes) to calculate the size of the metadata area.

Normally, you can let the system calculate the metadata size for you. If you want to estimate the size of the metadata area, see the chapter on table performance considerations in the *HCL OneDB™ Performance Guide*.

Figure 29: A Single Sbspace Chunk on page 308 illustrates the chunk structure of an sbspace as it appears immediately after the sbspace is created. Each reserved area can be allocated to either the user-data or metadata area. Reserved areas are always within the user-data area of the chunk.

Figure 29. A Single Sbspace Chunk



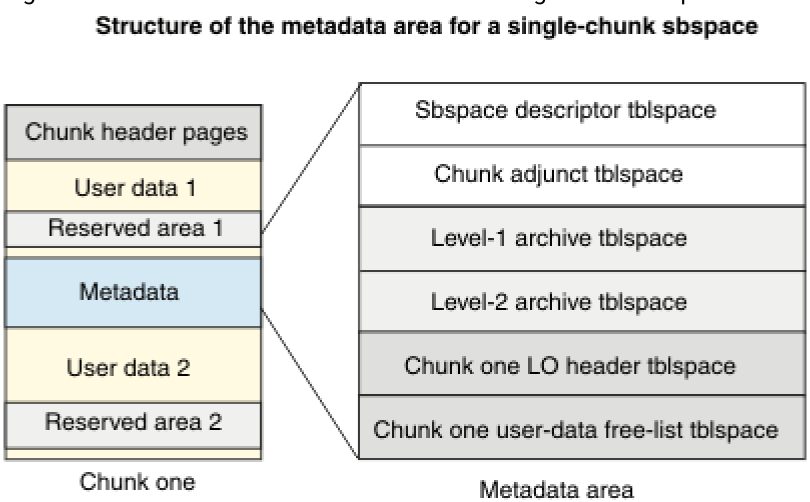
Because the chunk in Figure 29: A Single Sbspace Chunk on page 308 is the first in the sbspace, it contains an sbspace descriptor. The chunk descriptor tblspace in **chunk one** contains information about chunk one and all chunks added to the sbspace thereafter.

Structure of the metadata area

An sbspace contains a metadata area for each chunk in the sbspace.

As with the chunk header pages, four areas are exclusive to the first chunk in a sbspace: the sbspace descriptor tblspace, the chunk adjunct tblspace, and the level-1 and level-2 archive tblspaces. The tblspace header section contains a tblspace header for each of these tblspaces (notably excluding the tblspace **tblspace**). Figure 30: Structure of the metadata area for a single-chunk sbspace on page 308 shows the layout of the metadata in the single-chunk sbspace.

Figure 30. Structure of the metadata area for a single-chunk sbspace



When you specify the sbspace name in the **oncheck -ps** option, you can display the number of pages allocated and used for each tblspace in the metadata area.

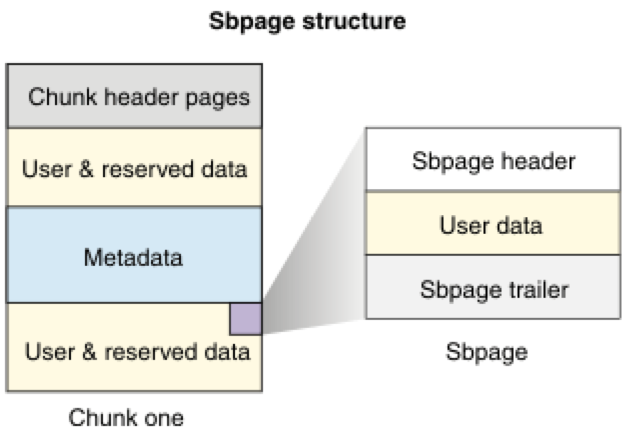
The following items describe how the metadata area grows:

- The sbspace descriptor tblspace does not grow.
- The chunk adjunct tblspace grows as chunks are added.
- The LO header tblspace grows as large objects are added to the chunk.
- The tblspace for user-data free list grows if free spaces in the chunk are heavily fragmented.

Sbpage Structure

Each sbpage is composed of three elements: an sbpage header, the actual user data itself, and an sbpage trailer. Figure 31: Sbpage Structure on page 309 shows the structure of an sbpage. The sbpage header consists of the standard page header. The sbpage trailer is used to detect an incomplete write on the page and to detect page corruption.

Figure 31. Sbpage Structure



Time Stamps

The database server uses a time stamp to identify a time when an event occurred relative to other events of the same kind. The time stamp is not a literal time that refers to a specific hour, minute, or second. It is a 4-byte integer that the database server assigns sequentially.

Database and Table Creation: What Happens on Disk

This section explains how the database server stores data related to the creation of a database or table and allocates the disk structures that are necessary to store your data.

Database Creation

After the root dbspace exists, users can create a database. The paragraphs that follow describe the major events that occur on disk when the database server adds a new database.

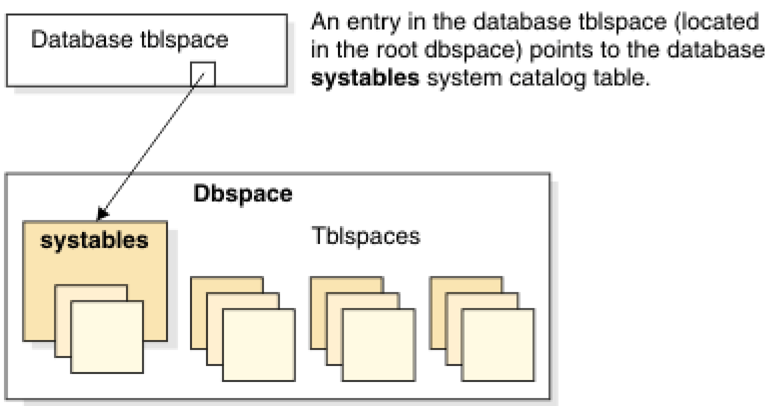
Disk-Space Allocation for System Catalog Tables

The database server searches the chunk free-list pages in the dbspace, looking for free space in which to create the system catalog tables. For each system catalog table, in turn, the database server allocates eight contiguous pages, the size of the initial extent of each system catalog table. The tables are created individually and do not necessarily reside next to each other in the dbspace. They can be located in different chunks. As adequate space is found for the initial extent of each table, the pages are allocated, and the associated chunk free-list page is updated.

Tracking of System Catalog Tables

The database server tracks newly created databases in the database tblspace, which resides in the root dbspace. An entry describing the database is added to the database tblspace in the root dbspace. (See Structure of the Database Tblspace on page 289.) For each system catalog table, the database server adds a one-page entry to the tblspace **tblspace** in the dbspace where the database was built. (See Structure of the Tblspace Tblspace on page 286.) Figure 32: New Databases on page 310 illustrates the relationship between the database tblspace entry and the location of the **systables** system catalog table for the database.

Figure 32. New Databases



For instructions on how to list your databases after you create them, see monitoring databases in the chapter on managing database-logging status in the *HCL OneDB™ Administrator's Guide*.

Table Creation

After the root dbspace exists, and a database has been created, users with the necessary SQL privileges can create a database table. When users create a table, the database server allocates disk space for the table in units called extents (see what is an extent in the chapter on where data is stored in the *HCL OneDB™ Administrator's Guide*). The paragraphs that follow describe the major events that occur when the database server creates a table and allocates the initial extent of disk space.

Disk-Space Allocation

The database server searches the chunk free-list pages in the dbspace for contiguous free space equal to the initial extent size for the table. When adequate space is found, the pages are allocated, and the associated chunk free-list page is updated.

If the database server cannot find adequate contiguous space anywhere in the dbspace, it allocates to the table the largest available amount of contiguous space. No error message is returned if an allocation is possible, even when the amount of space allocated is less than the requested amount. If the minimum extent size cannot be allocated, an error is returned. (Extents cannot span two chunks.)

Entry in the Tblspace Tblspace

The database server adds a one-page entry for this table to the tblspace **tblspace** in this dbspace. The tblspace number assigned to this table is derived from the logical page number in the tblspace **tblspace** where the table is described. See Tblspace Numbers on page 287.

The tblspace number indicates the dbspace where the tblspace is located. Tblspace extents can be located in any of the dbspace chunks.

If you must know exactly where the tblspace extents are located, execute the **oncheck -pe** command for a listing of the dbspace layout by chunk.

Entries in the System Catalog Tables

The table itself is fully described in entries stored in the system catalog tables for the database. Each table is assigned a table identification number or *tabid*. The tabid value of the first user-defined table object in a database is always `100`. (The object whose tabid = 100 might also be a view, synonym, or a sequence.) For a complete discussion of the system catalog, see the *HCL OneDB™ Guide to SQL: Reference*.

A table can be located in a dbspace that is different than the dbspace that contains the database. The tblspace itself is the sum of allocated extents, not a single, contiguous allocation of space. The database server tracks tblspaces independently of the database.

Creation of a Temporary Table

The tasks involved in creating temporary tables are similar to the tasks that the database server performs when it adds a new permanent table. The key difference is that temporary tables do not receive an entry in the system catalog for the database. For more information, see the section defining a temporary table, in the chapter on where data is stored in the *HCL OneDB™ Administrator's Guide*.

Administrative Utilities

Overview of Utilities

The HCL OneDB™ database server utilities allow you to perform administrative tasks directly from the command line.

The database server utilities support multibyte command-line arguments. For a complete list of the utilities that support multibyte command-line arguments, see the Locale-specific support for utilities.

The database server must be online before you execute a utility, with the following exceptions:

- **oninit**
- Some **onlog** options
- Some **oncheck** options

Note: When using utilities, do not use the UNIX™ command `CTRL-C` to send an interrupt signal to a process because it might produce an error.

Obtaining utility version information

Use the **-V** and **-version** options of many HCL OneDB™ command-line utilities to obtain version, primarily for debugging.

About this task

The **-V** option displays the software version number and the serial number.

The **-version** option extends the **-V** option to display additional information about the build operation system, build number, and build date.

utility { *utility specific options* **| -V | -version** }

The **-V** and **-version** options cannot be used with any other utility options. For example, the **onstat -version** command might display the following output.

```
onstat -version

        Program:         onstat
    Build Version:    11.70.FC1
    Build Host:       connla
    Build OS:         SunOS 5.6
    Build Number:     009
    Build Date:       Sat Nov 20 03:38:27 CDT 2011
    GLS Version:      glslib-4.50.xC2
```

The **onstat -V** command might display the following information:

```
IBM Informix Version 11.70.FC1   Software Serial Number
 RDS#N000000
```

Setting local environment variables for utilities

On UNIX™ operating systems, you can start certain utilities without setting local environment variables in your shell environment. You can set local environment variables in the `onconfig` file. When you run the command to start the utility, use the -FILE option to point to the `onconfig` file.

Before you begin

Before you begin, ensure that these prerequisites are met:

- The path to the executable program for the utility is part of the existing shell environment.
- If you want to run commands on a remote computer, a remote shell utility such as SSH is configured.

About this task

1. Add values for one or more environment variables to the `onconfig` file. Use the following format for each directive:

   Example

   #$variable_name value
2. When you run the command to start the utility, use the -FILE option to specify the full or relative path to the `onconfig` file.

   Review the syntax, usage, and examples in the reference information for the -FILE option.

Results

The utility reads and sets the environment variables that are specified in the `onconfig` file, and those values take precedence over values that are set in the local shell environment.

The finderr utility

Use the finderr utility to view additional information on HCL OneDB™ error messages. On UNIX™ and Linux™ platforms, the information appears on the command line. On Windows™ platforms, the information appears in the Error Messages program.

Syntax

```
finderr [{ - | + }] error_number
```

Table 114. finderr element

Element	Purpose	Key Considerations
error_number	The error message number for which to provide additional information	**On UNIX™ or Linux™:** If you do not include a minus sign (-) or plus sign (+) and both a positive and a negative version of the error message exists, the negative version of the message is displayed. To display the information about an error message number that is positive, preface the error number with a plus sign. **On Windows™:** If you do not include a minus sign or plus sign and both a positive and a negative version of the error message exists, you must choose which message you want to view in the Error Messages program.

Usage

Error messages that are printed in the message log include a message number and a short message description. Use the message number with the finderr command to look up a more detailed description of the cause of the error and possible user actions to correct or prevent the error.

On Windows™, you can open the Error Messages program directly by choosing **Error Messages** from the database server program group.

Example

Examples

The following command on a UNIX™ or Linux™ platform displays information about the error message -201:

```
finderr 201

-201  A syntax error has occurred.

This general error message indicates mistakes in the form of an SQL
statement. Look for missing or extra punctuation (such as missing or
extra commas, omission of parentheses around a subquery, and so on),
keywords misspelled (such as VALEUS for VALUES), keywords misused (such
as SET in an INSERT statement or INTO in a subquery), keywords out of
sequence (such as a condition of "value IS NOT" instead of "NOT value
IS"), or a reserved word used as an identifier.

Database servers that provide full NIST compliance do not reserve any
words; queries that work with these database servers might fail and
return error -201 when they are used with earlier versions of IBM Informix
database servers.

The cause of this error might be an attempt to use round-robin syntax with
CREATE INDEX or ALTER FRAGMENT INIT on an index. You cannot use round-robin
indexes.

The error may also occur if an SQL statement uses double quotation marks
around input strings and the environment variable DELIMIDENT is set.
If DELIMIDENT is set, strings that are surrounded by double quotation
marks are regarded as SQL identifiers rather than string literals. For
more information on the usage of DELIMIDENT, see the IBM Informix Guide to
SQL: Reference.
```

The following command displays information about the error message 100, which corresponds to the SQLCODE value of 100:

```
finderr +100

100 No matching records found.

The database server did not find any more data. This message is an ANSI-
standard SQLCODE value. If you attempted to select or fetch data, you
encountered the end of the data, or no data matched the criteria in the
WHERE clause. Check for an empty table. Use this SQLCODE value to determine
when a statement reaches the end of the data. For more information, see the
discussion of SQLCODE in the IBM Informix ESQL/C Programmer's Manual. The
database server can return this SQLCODE value to a running program.
```

The genoncfg Utility

Use the **genoncfg** utility to expedite the process of customizing the default HCL OneDB™ configuration file (**onconfig.std**) to the host environment and your planned usage of a database server instance.

Syntax

genoncfg { *input_file [ONEDB_HOME]* | **-h** | **-V** | **-version** }

Element	Purpose	Key Considerations
input_file	Name of the input file containing your parameter settings.	
ONEDB_HOME	Path to the HCL OneDB™ installation that you want to configure.	You can omit the installation path if the **ONEDB_HOME** environment variable is set. If the **ONEDB_HOME** variable is already set and you enter an installation path on the command line, the utility runs with the command-line path.
-h	Help information about the **genoncfg** utility.	
-V	Displays short version information and exits the command-line utility.	
-version	Displays extended version information and exits the command-line utility.	

Usage

Log in to the host computer as root or user **informix** before you run this utility.

You must set parameters that are valid for your host environment in an input file before you can successfully run the **genoncfg** utility. For all environments, the parameter `disk` is required in the input file. You can also enter directives in the input file. The directives are not required to run the utility, but they can be helpful in some circumstances.

The utility does not read or modify any existing configuration file. If you have a pre-existing ONCONFIG file in the host environment, none of its parameter values are changed when you run the utility. Therefore, you can review the recommended configuration settings before you put them in effect on a database server instance.

To use the genoncfg utility:

1. Create the input file containing your values for the parameters that the **genoncfg** utility processes with a text editor.
2. Run the utility with your input file. The configuration file (named **onconfig**) is generated and saved in the working directory.
3. *Optional:* Rename the generated configuration file.
4. If you want to run a database server instance with the generated configuration file, copy the file to `$ONEDB_HOME/etc` and update the **ONCONFIG** environment variable accordingly.

Input File for the genoncfg Utility

Use the input file to specify the following information about the database server instance:

- number of anticipated online transaction processing (OLTP) connections
- number of anticipated decision-support systems (DSS) connections

- disk space
- CPU utilization
- network connection settings
- recovery time

The input file is an ASCII text file. There is no required order for the parameters. The following is an example of an input file:

```
cpus 1
memory 1024 m
connection name demo_on onsoctcp 9088
servernum 1
oltp_connections 10
dss_connections 2
disk /opt/IBM/informix/demo/server/online_root 0 k 300 m
directive one_crit
directive debug
```

Table 115. Parameters of the Input File for the genoncfg Utility

Parameter	Description
connection	Server connection parameters: • `name` or `alias`, depending on whether the connection functions with a specific server name (the DBSERVERNAME parameter of the configuration file) or with an alternative server name (using the DBSERVERALIASES parameter of the configuration file) • name for the connection • type of server connection (equivalent to NETTYPE in the configuration file) • port number for the service Example: `connection name demo_on onsoctcp 9088`
cpus	Number of central processing units (CPUs) to allocate the instance. Example: `cpus 1`
directive	Directives that can be used with the **genoncfg** utility. • one_crit: Configures the database server to store physical logs, logical logs, and data in the root dbspace only. • debug: Displays information in real time about the host environment and actions done on the configuration file. Example: `directive one_crit` This information can be helpful in troubleshooting problems with database server configuration. One scenario is that the debug directive can result in saving time. In this scenario, you read the displayed information and notice that

Table 115. Parameters of the Input File for the genoncfg Utility (continued)

Parameter	Description
	the utility is creating an onconfig file that you do not want or that will not function. You stop the utility while it is still running, adjust the input file settings, and then rerun the utility with the modified input file.
disk	Disk storage space settings for the instance: • location of the root dbspace • size of offset, in megabytes (m) or kilobytes (k) • size of root dbspace, in megabytes (m) or kilobytes (k) Example: UNIX™: `/opt/IBM/dbspace/rootdbs` Windows™: `d:\INFXDATA\rootdbs` **Important:** If you enter a path location that is the root dbspace of a working instance, the instance is overwritten and made unusable.
dss_connections	Estimated number of decision-support systems (DSS) connections to the instance. For example, a query client or other application that obtains result sets for business intelligence can be a DSS connection. Example: `dss_connections 2`
memory	Amount of memory, in megabytes (m), for the instance. Example: `memory 1024 m`
oltp_connections	Estimated number of online transaction processing (OLTP) connections to the instance. Typically, an application that modifies the state of databases in the instance is an OLTP connection. Example: `oltp_connections 10`
rto_server_restart	Specifies the amount of time, in seconds, that the database server has to recover from a problem after you restart HCL OneDB™ and bring it into online or quiescent mode. The value can be set either to 0 to disable the configuration parameter or to a value between 60 and 1800 to enable the parameter and indicate the number of seconds. Example: `rto_server_restart 100` specifies the recovery time objective as 100 seconds.

Table 115. Parameters of the Input File for the genoncfg Utility (continued)

Parameter	Description
servernum	Unique ID of the database server instance. Example: `servernum 1`

The oncheck Utility

Use the oncheck utility to check specified disk structures for inconsistencies, repair inconsistent index structures, and display information about disk structures.

The oncheck utility requires sort space when examining an index. The amount of sort space required is the same as that needed to build the index. For information about calculating the amount of temporary space needed, see Estimating temporary space for index builds on page . If you receive the error "`no free disk space for sort`," you must estimate the amount of temporary space needed and make that space available.

You can use SQL administration API commands that are equivalent to some oncheck commands.

oncheck Check-and-Repair

The oncheck utility repairs disk structures.

The oncheck utility can repair the following types of disk structures:

- Partition page statistics
- Bitmap pages
- Partition blobpages
- Blobspace blobpages
- Indexes
- Sbspace pages
- Metadata partitions for sbspaces

If oncheck detects inconsistencies in other structures, messages alert you to these inconsistencies, but oncheck cannot resolve the problem. For more information, see the chapter on consistency checking in the *HCL OneDB™ Administrator's Guide* and Disk Structures and Storage on page 282.

What Does Each Option Do?

The oncheck options fall into three categories: check, repair, and display.

The display or print options (those prefixed with the letter **p**) are identical in function to the **-c** options, except that the **-p** options display additional information about the data that is being checked as the oncheck utility executes. You cannot combine oncheck option flags except as the following paragraphs describe.

In general, the **-c** options check for consistency and display a message on the screen only if they find an error or inconsistency.

Any user can execute the check options. On UNIX™ platforms, you must be user **informix** or **root** to display database data or initiate repair options. On Windows™, you must be a member of the **Informix-Admin** group to display database data or initiate repair options.

Table 116: oncheck Options and Their Function on page 319 associates oncheck options with their function. It also shows the SQL administration API *command* strings that are equivalent to the **oncheck -c** options.

Table 116. oncheck Options and Their Function

Object	Check	SQL administration API *command* string	Repair	Display
Blobspace simple large objects				**-pB**
System catalog tables	**-cc**			**-pc**
Data rows, no simple large objects or smart large objects	**-cd**			**-pd**
Data rows, simple large objects but no smart large objects	**-cD**			**-pD**
Table with a user-defined access method	**-cd, -cD**	CHECK DATA		
Chunks and extents	**-ce**	CHECK EXTENTS		**-pe**
Index (key values)	**-ci, -cix**		**-ci -y -pk -y, -pkx -y**	**-pk**
Index (keys plus rowids)	**-cI, -cIx**		**-cI -y -pK -y, -pKx -y**	**-pK**
Index with a user-defined access method	**-ci, -cI**			
Index (leaf key values)			**-pl -y, -plx -y**	**-pl**
Index (leaf keys plus rowids)			**-pL -y, -pLx -y**	**-pL**
Pages (by table or fragment)				**-pp**
Pages (by chunk)				**-pP**
Root reserved pages	**-cr, -cR**			**-pr, -pR**
Metadata for smart large objects	**-cs, -cS**			**-ps, -pS**
Space usage (by table or fragment)		CHECK PARTITION PRINT PARTITION		**-pt**

Table 116. oncheck Options and Their Function (continued)

Object	Check	SQL administration API *command* string	Repair	Display
Space usage (by table, with indexes)				**-pT**

Using the -y Option to Perform Repairs

Use the **-y** option to instruct oncheck to perform repairs automatically.

If you do not use the **-y** option, oncheck prompts you when it encounters an inconsistency and allows you to request a repair. If you specify option **-n**, oncheck does not prompt you because this option instructs oncheck to not perform repairs.

The following examples show automatic repair commands for the oncheck utility:

```
oncheck -cd -y
oncheck -cD -y
oncheck -ci -y
oncheck -cI -y
```

Repairing Indexes in Sbspaces and External Spaces

The **oncheck** utility can repair an index in an sbspace or external space if the index is created using an access method that supports the **oncheck -y** option.

Although the **oncheck** utility does not repair fragmented indexes, user-defined access methods can repair them. For more information about the **oncheck** options that access methods support, see the *HCL OneDB™ DataBlade® API Programmer's Guide* or the *HCL OneDB™ Virtual-Index Interface Programmer's Guide*.

Locking and oncheck

The oncheck utility places a shared lock on a table, so no other users can perform updates, inserts, or deletes until the check has completed.

The oncheck utility places a shared lock on a table during the following operations:

- When it checks data
- When it checks indexes (with **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL**) and the table uses page locking
- When you specify the **-x** option with **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** and the table uses row locking

If the table does not use page locking, the database server does not place a shared lock on the table when you check an index with the **oncheck -ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** options. When no shared lock is on the table during an index check, other users can update rows during the check.

By not placing a shared lock on tables using row locks during index checks, the oncheck utility cannot be as accurate in the index check. For absolute assurance of a complete index check, you can execute oncheck with the **-x** option. With the **-x** option, oncheck places a shared lock on the table, and no other users can perform updates, inserts, or deletes until the check has completed.

The oncheck utility returns unreliable results when run on secondary servers in a high-availability cluster.

For more information about the **-x** option, refer to Turn On Locking with -x on page 342. For information on shared locks and intent shared locks, see the *HCL OneDB™ Performance Guide*.

The oncheck utility places a shared lock on system catalog tables when they are checked. It places an exclusive lock on a table when it executes repair options.

oncheck utility syntax

The **oncheck** utility checks specified disk structures for inconsistencies, repairs inconsistent index structures, and displays information about disk structures.

```
>>-oncheck--+-------------------------+------------------------>
            |                     (1) |
            '-|  -FILE option   |-----'

>--+-+- -ce-+-------------------------------------------------------------------+-->
   | +- -pe-+                                                                   |
   | +- -cr-+                                                                   |
   | +- -pr-+                                                                   |
   | +- -cR-+                                                                   |
   | '- -pR-'                                                                   |
   +-+- -ci-+--+-----+--database--+---------------------------------------+-----+
   | +- -cl-+  '- -x-'            '-:--+--------+--table--+-------------+-'     |
   | +- -pk-+                          '-owner.-'         '-#index_name-'       |
   | +- -pK-+                                                                   |
   | +- -pl-+                                                                   |
   | '- -pL-'                                                                   |
   +-+- -cd-+--database--+--------------------------------------+---------------+
   | '- -cD-'            '-:--+--------+--table--+------------+-'               |
   |                          '-owner.-'         +-,frag_dbs--+                 |
   |                                             '-%frag_part-'                 |
   +-+- -cc-+--+----------+-----------------------------------------------------+
   | '- -pc-'  '-database-'                                                     |
   +-+- -pB-+--database--+-------------------------------------+----------------+
   | +- -pt-+            '-:--+--------+--table--+-----------+-'                |
   | '- -pT-'                 '-owner.-'         '-,frag_dbs-'                  |
   +-+- -pd-+--+-database--+------------------------------------------------+-+-+
   | '- -pD-'  |           '-:--+--------+--table--+----------+--+-------+-'  | |
   |           |                '-owner.-'         '-,frag_dbs-'  '-rowid-'   | |
   |           '-tblspacenum--+-----------------+-----------------------------' |
   |                          '-logical pagenum-'                               |
   +- -pp--+-database--:--+--------+--table--+-----------+--rowid-+-------------+
   |       |              '-owner.-'         '-,frag_dbs-'        |             |
   |       '-tblspacenum--logical pagenum-------------------------'             |
   +- -pP--chunknum--logical pagenum--------------------------------------------+
   +-+- -cs-+--+---------+------------------------------------------------------+
   | '- -cS-'  '-sbspace-'                                                      |
   +-+- -ps-+--+---------------------------+------------------------------------+
   | '- -pS-'  '-sbspace--partnum--pagenum-'                                    |
   +- -u--option--+------------------+------------------------------------------+
```

```
   |              '-(--arg_string--)-'                                  |
   '-+-----------+------------------------------------------------------------'
     +- -V-------+
     '- -version-'

>--+-----+--+-----+--------------------------------------------><
   +- -n-+  '- -q-'
   '- -y-'
```

Note: See The -File Option on page 356

Element	Purpose	Key Considerations
-cc	Checks system catalog tables for the specified database	See oncheck -cc and-pc: Check system catalog tables on page 327.
-cd	Reads all pages except simple large objects from the tblspace for the specified database, table, or fragment and checks each page for consistency Also checks tables that use a user-defined access method	Does not check simple or smart large objects. See oncheck -cd and oncheck -cD commands: Check pages on page 327.
-cD	Same as -**cd** but also reads the header of each blobpage and checks it for consistency	Checks simple large objects but not smart large objects. See oncheck -cd and oncheck -cD commands: Check pages on page 327 .
-ce	Checks each chunk-free list and corresponding free space and each tblspace extent. Also checks smart-large-object extents and sbspace metadata	The **oncheck** process verifies that the extents on disk correspond to the current control information that describes them. See oncheck -ce, -pe: Check the chunk-free list on page 329. For background information, see Next-Extent Allocation on page 293.
-ci	Checks the ordering of key values and the consistency of horizontal and vertical node links for all indexes associated with the specified table Also checks indexes that use a user-defined access method	See oncheck -ci and -cI: Check index node links on page 329.

Element	Purpose	Key Considerations
-cI	Same as -**ci** but also checks that the key value tied to a rowid in an index is the same as the key value in the row	See oncheck -ci and -cI: Check index node links on page 329.
-cr	Checks each of the root dbspace reserved pages for several conditions	See oncheck -cr and -cR: Check reserved pages on page 331.
-cR	Checks the root dbspace reserved pages, physical-log pages, and logical-log pages	See oncheck -cr and -cR: Check reserved pages on page 331
-cs	Checks smart large object and sbspace metadata for an sbspace	See oncheck -cs, -cS, -ps, -pS: Check and display sbspaces on page 331.
-cS	Checks smart large object and sbspace metadata for an sbspace as well as extents	See oncheck -cs, -cS, -ps, -pS: Check and display sbspaces on page 331.
sbspace	Indicates optional sbspace name If not supplied, all sbspaces are checked.	None.
-n	Indicates that no index repair should be performed, even if errors are detected	Use with the index repair options (**-ci**, **-cI**, **-pk**, **-pK**, **-pl**, and **-pL**).
-pB	Displays statistics that describe the average fullness of blobspace blobpages in a specified table	These statistics provide a measure of storage efficiency for individual simple large objects in a database or table. If a table or fragment is not specified, statistics are displayed for the entire database. See oncheck -pB: Display blobspace statistics on page 332. For information about optimizing blobspace blobpage size, see the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-pc	Same as **-cc** but also displays the system catalog information as it checks the system catalog tables, including extent use for each table	None.
-pd	Displays rows in hexadecimal format	See oncheck -pd and pD: Display rows in hexadecimal format on page 332.
-pD	Displays rows in hexadecimal format and simple-large-object values stored in the tblspace or header information for smart large objects stored in an sbspace	See oncheck -pd and pD: Display rows in hexadecimal format on page 332.

Element	Purpose	Key Considerations
	sbpage and simple large objects stored in a blobspace blobpage	
-pe	Same as **-ce** but also displays the chunk and tblspace extent information as it checks the chunk free list, the corresponding free space, and each tblspace extent	See oncheck -ce, -pe: Check the chunk-free list on page 329.
-pk	Same as **-ci** but also displays the key values for all indexes on the specified table as it checks them	See oncheck -pk, -pK, -pl, -pL: Display index information on page 334.
-pK	Same as **-cI** but also displays the key values and rowids as it checks them	See oncheck -pk, -pK, -pl, -pL: Display index information on page 334.
-pl	Same as **-ci** but also displays the key values. Only leaf-node index pages are checked	See oncheck -pk, -pK, -pl, -pL: Display index information on page 334.
-pL	Same as **-cI** but also displays the key values and rowids for leaf-node index pages only	See oncheck -pk, -pK, -pl, -pL: Display index information on page 334.
-pp	Displays contents of a logical page	See oncheck -pp and -pP: Display the contents of a logical page on page 335.
-pP	Same as **-pp** but requires a chunk number and logical page number or internal rowid as input	See oncheck -pp and -pP: Display the contents of a logical page on page 335.
-pr	Same as **-cr** but also displays the reserved-page information as it checks the reserved pages	See oncheck -pr and pR: Display reserved-page information on page 337.
-pR	Same as **-cR** but also displays the information for the reserved pages, physical-log pages, and logical-log pages	See oncheck -pr and pR: Display reserved-page information on page 337.
-ps	Checks and displays smart-large-object and sbspace metadata for an sbspace	See oncheck -cs, -cS, -ps, -pS: Check and display sbspaces on page 331.
-pS	Checks and displays smart-large-object and sbspace metadata. Lists extents and header information for individual smart large objects	See oncheck -cs, -cS, -ps, -pS: Check and display sbspaces on page 331.

Element	Purpose	Key Considerations
-pt	Displays tblspace information for a table or fragment	See oncheck -pt and -pT: Display tblspaces for a Table or Fragment on page 339.
-pT	Same as **-pt** but also displays index-specific information and page-allocation information by page type (for dbspaces)	See oncheck -pt and -pT: Display tblspaces for a Table or Fragment on page 339.
-q	Suppresses all checking and validation message	None.
-x	Places a shared lock on the table when you check and print an index	Use with the **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, or **-pL** options. For complete information, see Turn On Locking with -x on page 342.
-y	Repairs indexes when errors are detected	None.
-V	Displays the software version number and the serial number	See Obtaining utility version information on page 312.
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See Obtaining utility version information on page 312.
chunknum	Specifies a decimal value that you use to indicate a particular chunk	Value must be an unsigned integer greater than 0. Chunk must exist. Execute the **-pe** option to learn which chunk numbers are associated with specific dbspaces, blobspaces or sbspaces.
database	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
db1	Specifies the local database that contains a data type that you want to check	Optionally specify the local database server name using the format **db1@server1**.
db2	Specifies the remote database that contains a data type that you want to check	Optionally specify the remote database server name using the format **db2@server2**.
frag_dbs	Specifies the name of a dbspace that contains a fragment you want to check for consistency	Dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.

Element	Purpose	Key Considerations
index_name	Specifies the name of the index that you want to check for consistency	Index must exist on table and in database specified. Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
logical pagenum	Specifies an integer value that you use to indicate a particular page in a tblspace	Value must be an unsigned integer between 0 and 16,777,215, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
object	Specifies the name of the DataBlade®, cast, operator , user-defined data type, or UDR that you want to check	If you do not specify an object name, the database server compares all objects of the same type with the same name and owner.
owner	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; for more information, see *HCL OneDB™ Guide to SQL: Syntax*.
pagenum	Indicates the page number of the sbspace metadata portion to check and display	None.
partnum	Identifies the sbspace metadata partition to check and display	None.
rowid	Identifies the rowid of the row whose contents you want to display. The rowid is displayed as part of **oncheck -pD** output	Value must be an unsigned integer between 0 and 4,277,659,295, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
sbspace	Specifies the name of the sbspace that you want to check for consistency	None.
server	Specifies the database server name	If you omit the database server name, **oncheck** uses the name that ONEDB_SERVER specifies.
table	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; for more information, see *HCL OneDB™ Guide to SQL: Syntax*.
tblspacenum	Identifies the tblspace whose contents you want to display	Value must be an unsigned integer between 0 and 208,666,624, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.

oncheck -cc and-pc: Check system catalog tables

The **-cc** option checks system catalog tables for information about database tables, columns, indexes, views, constraints, stored procedures, and privileges.

```
>>-oncheck----+- -cc-+--database------------------------------><
              '- -pc-'
```

The oncheck -cc command checks the following tables:

- **systables**
- **syscolumns**
- **sysindices**
- **systabauth**
- **syscolauth**
- **sysdepend**
- **syssyntable**
- **sysviews**
- **sysconstraints**
- **sysams**

If you do not specify a database name in the oncheck -cc, the command checks the listed system catalog tables for all databases.

The **-pc** option performs the same checks on system catalog tables and also displays the system catalog information, including the physical address, type of locking used, row size, number of keys, extent use, the number of pages allocated and used, tblspace partnum, and index use for each table.

Before you execute oncheck -cc or oncheck -pc, execute the SQL statement UPDATE STATISTICS to ensure that an accurate check occurs. To check a table, oncheck compares each system catalog table to its corresponding entry in the tblspace.

oncheck -cd and oncheck -cD commands: Check pages

Use the oncheck -cd and oncheck -cD commands to check each page for consistency. Use the oncheck -cd -y or oncheck -cD -y command to repair inconsistencies.

```
>>-oncheck------------------------------------------------------>

>----+- -cd-+--+-----+--database--+--------------------------------------+---><
     '- -cD-'  '- -y-'            '-:--+--------+--table--+------------+-'
                                       '-owner.-'         +-,frag_dbs--+
                                                          '-%frag_part-'
```

The oncheck -cd command reads all pages, except for blobpages and sbpages, from the tblspace for the specified database, table, fragment, or multiple fragments (fragparts), and checks each page for consistency. This command compares entries in the bitmap page to the pages to verify mapping.

The oncheck -cD command performs the same checks as the oncheck -cd command, and also checks the header of each blobpage for consistency. The oncheck -cD command does not compare the beginning time stamps stored in the header with the ending time stamps stored at the end of a blobpage. Use the oncheck -cD -y command to clean up orphaned simple large objects in blobspaces, which can occur after a rollback across several log files.

If the database contains fragmented tables, but no fragment is specified, the oncheck -cd command checks all fragments in the table. If you do not specify a table, the command checks all of the tables in the database. By comparison, the oncheck -pd command displays a hexadecimal dump of specified pages but does not check for consistency.

For both the oncheck -cd and oncheck -cD commands, the oncheck utility locks each table as it checks the indexes for the table. To repair the pages, use oncheck -cd -y or oncheck -cD -y.

If tables are fragmented on multiple partitions in the same dbspace, the oncheck -cd and oncheck -cD commands show the partition names. The following example shows typical output for a table that has fragments in multiple partitions in the same dbspace:

```
TBLspace data check for multipart:informix.t1
              Table fragment partition part_1 in DBspace dbs1
              Table fragment partition part_2 in DBspace dbs1
              Table fragment partition part_3 in DBspace dbs1
              Table fragment partition part_4 in DBspace dbs1
              Table fragment partition part_5 in DBspace dbs1
```

When you use the oncheck -cd or oncheck -cD command, you can specify either the *frag_dbs* or the *%frag_dbs* option but not both:

- When you use the *frag_dbs* option, the utility checks all fragments in the dbspace `frag_dbs`.
- When you use the *%frag_dbs* option, the utility checks only the fragment named *frag_part*, if the PARTITION syntax was used when the fragment or table was created.

While it is possible to fragment an index with the PARTITION syntax, it is not possible to limit an index check to just one fragment or partition. For example, you can specify oncheck -cDI my_db:my_tab,data_dbs1 or oncheck -cDI my_db:my_tab %part1. The D (data) portion of the check is limited according to the specification, however the I (index) check is not limited.

Examples

The following example checks the data rows, including simple large objects and smart large objects, in the **catalog** table:

```
oncheck -cD superstores_demo:catalog
```

If you specify a single fragment, the oncheck utility displays a single header for that fragment. For fragmented tables, one header is displayed for each fragment:

```
TBLspace data check for stores_demo:informix.tab1
 Table fragment in DBspace db1
```

Messages

If the oncheck utility finds no inconsistencies, a header displays for each table that the utility. For example:

```
TBLSPACE data check for stores_demo:informix.customer
```

If the oncheck utility finds an inconsistency, a message displays. For example:

```
BAD PAGE 2:28: pg_addr 2:28 != bp-> bf_pagenum 2:69
```

The physical address `2:28` represents page 28 of chunk number 2.

If an index that uses an access method provided by a DataBlade® module cannot find the access method, you receive the following message:

```
-9845 Access method access_method_name does not exist in database.
Ensure that the DataBlade installation was successful.
```

Reference

To monitor blobspace blobpages, see oncheck -pB: Display blobspace statistics on page 332.

oncheck -ce, -pe: Check the chunk-free list

```
>>-oncheck----+- -ce-+-----------------------------------------><
              '- -pe-'
```

The **-ce** option checks each chunk-free list and corresponding free space and each tblspace extent. For more information, refer to Next-Extent Allocation on page 293 and Structure of the Chunk Free-List Page on page 285, respectively. The **oncheck** process verifies that the extents on disk correspond to the current control information that describes them.

The **-pe** option performs the same checks and also displays the chunk and tblspace extent information during the check. The **-ce** and **-pe** options also check blobspaces, smart-large-object extents, and user-data and metadata information in sbspace chunks.

For information about using **oncheck -ce** and **-pe**, see managing disk space in the *HCL OneDB™ Administrator's Guide*.

Use CHECK EXTENTS as the SQL administration API *command* string for **oncheck -ce**.

oncheck -ci and -cI: Check index node links

Use the oncheck -ci and oncheck -cI commands to check the ordering of key values and the consistency of horizontal and vertical node links for all indexes associated with the specified table.

The oncheck -cI command also checks that the key value tied to a rowid in an index is the same as the key value in the row. The **-cI** option does not cross-check data on a functional index.

```
>>-oncheck------------------------------------------------------>

>----+- -ci +--database--+-------------------------------------+---><
     '- -cI '            '-:--+--------+--table--+----------+-'
                              '-owner.-'         '- #index -'
```

If you do not specify an index, the option checks all indexes. If you do not specify a table, the option checks all tables in the database.

The same **-ci** repair options are available with **-cI**. If **oncheck -ci** or **oncheck -cI** detects inconsistencies, it prompts you for confirmation to repair the problem index. If you specify the **-y** (yes) option, indexes are automatically repaired. If you specify the **-n** (no) option, the problem is reported but not repaired; no prompting occurs.

If **oncheck** does not find inconsistencies, the following message appears:

```
validating indexes......
```

The message displays the names of the indexes that **oncheck** is checking.

Note: Using **oncheck** to rebuild indexes can be time consuming. Processing is usually faster if you use the SQL statements DROP INDEX and CREATE INDEX to drop and re-create the index.

The following example checks all indexes on the **customer** table:

```
oncheck -cI -n stores_demo:customer
```

The following example checks the index **zip_ix** on the **customer** table:

```
oncheck -cI -n stores_demo:customer#zip_ix
```

If indexes are fragmented on multiple partitions in the same dbspace, the oncheck -ci and oncheck -cI commands show the partition names. The following example show typical output for an index that has fragments in multiple partitions in the same dbspace:

```
Validating indexes for multipart:informix.t1...
                Index idx_t1
                  Index  fragment partition part_1 in DBspace dbs1
                  Index  fragment partition part_2 in DBspace dbs1
                  Index  fragment partition part_3 in DBspace dbs1
                  Index  fragment partition part_4 in DBspace dbs1
                  Index  fragment partition part_5 in DBspace dbs1
```

By default, the database server does not place a shared lock on the table when you check an index with the oncheck -ci or oncheck -cI commands unless the table uses page locking. For absolute assurance of a complete index check, you can execute oncheck -cior oncheck -cI with the **-x** option. With the **-x** option, **oncheck** places a shared lock on the table, and no other users can perform updates, inserts, or deletes until the check has completed. For more information about using oncheck -ci and oncheck -cI with the **-x** option, Turn On Locking with -x on page 342.

When you execute **oncheck** on an external index, the user-defined access method is responsible for checking and repairing an index. If an index that employs a user-defined access method cannot find the access method, the database server reports an error. The **oncheck** utility does not repair inconsistencies in external indexes. You should not use **oncheck -cI** on a table that contains more than one type of index.

The **oncheck** utility requires sort space when examining an index. The amount of sort space required is the same as that needed to build the index. For information about calculating the amount of temporary space needed, see Estimating temporary space for index builds on page . If you receive the error "`no free disk space for sort`," you must estimate the amount of temporary space needed and make that space available.

For more information about indexes, see Structure of B-Tree Index Pages on page 299.

oncheck -cr and -cR: Check reserved pages

```
>>-oncheck----+- -cr-+-----------------------------------------><
              '- -cR-'
```

The -cr option checks each of the root dbspace reserved pages as follows:

- It validates the contents of the ONCONFIG file with the PAGE_CONFIG reserved page.
- It ensures that all chunks can be opened, that chunks do not overlap, and that chunk sizes are correct.

The -cR option performs the same checking and validation, and also checks all logical-log and physical-log pages for consistency. The -cr option is considerably faster because it does not check the log-file pages.

If you have changed the value of a configuration parameter (either through onparams, onmonitor, onspaces, or by editing the configuration file), but you have not yet reinitialized shared memory, oncheck -cr and oncheck -cR detect the inconsistency and return an error message.

If you have changed the value of a configuration parameter (either through ISA, onparams, onmonitor, onspaces, or by editing the configuration file), but you have not yet reinitialized shared memory, oncheck -cr and oncheck -cR detect the inconsistency and return an error message.

If oncheck -cr does not display any error messages after you execute it, you can assume that all three items in the preceding list were checked successfully.

For more information on reserved pages, see Reserved Pages on page 283.

oncheck -cs, -cS, -ps, -pS: Check and display sbspaces

```
>>-oncheck--+-+- -cs-+--+---------+-------------------+--------><
            | '- -cS-'  '-sbspace-'                   |
            '-+- -ps-+--+---------------------------+-'
              '- -pS-'  '-sbspace--partnum--pagenum-'
```

The **-cs** option checks sbspaces. The **-ps** option checks sbspaces and extents.

The **-cS** option validates and displays metadata for an sbspace.

The **-ps** option checks sbspaces and extents. If you do not specify the sbspace name, these options check all sbspaces.

The **-pS** option validates and displays metadata for an sbspace and also lists extents and header information for smart large objects.

If you do not specify the sbspace name, all sbspaces will be checked. The following example checks and displays metadata for **test_sbspace**:

```
oncheck -ps test_sbspace
```

If you specify **rootdbs** as the sbspace name with the **-cs** or **-ps** options, **oncheck** checks the root dbspace.

For more information about using the **-cs, -cS, -ps,** and**-pS** options, see the *.HCL OneDB™ Administrator's Guide*.

oncheck -pB: Display blobspace statistics

```
>>-oncheck---- -pB----database--+-------------------------------------+-><
                                '-:--+--------+--table--+-----------+-'
                                     '-owner.-'         '-,frag_dbs-'
```

The **-pB** option displays statistics that describe the average fullness of blobspace blobpages in a specified table. These statistics provide a measure of storage efficiency for individual simple large objects in a database or table. If you do not specify a table or fragment, the option displays statistics for the entire database. For more information, see optimizing blobspace blobpage size in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.

oncheck -pd and pD: Display rows in hexadecimal format

```
>>-oncheck------------------------------------------------------>

>----+- -pd-+--+-database--+-----------------------------------------------------------------+-+---><
     '- -pD-'  |           '-:--+--------+--table--+-----------+--+------------+--+-------+-' |
               |                '-owner.-'         '-,frag_dbs-'  '-,frag_part-'  '-rowid-'   |
               '-tblspacenum--+-----------------+---------------------------------------------'
                              '-logical pagenum-'
```

The **-pd** option takes a database, a table, a fragment, a fragment partition (fragpart), and a specific rowid or tblspace number and logical page number as input. In every case, **-pd** prints page-header information and displays the specified rows for the database object (database, table, fragment, internal rowid, or page number) that you specify in hexadecimal and ASCII format. No checks for consistency are performed.

Element	Purpose	Key Considerations
database	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
frag_dbs	Specifies the name of a dbspace that contains a fragment you want to check for consistency	Dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
frag_part	Specifies the fragment partition	For fragmented tables or an index that use expression-based or round-robin distribution schemes, you can create multiple partitions, which are collections of pages for a table or index, within a single dbspace. This partition is referred to as a *fragment partition* or *fragpart*.
logical pagenum	Specifies an integer value that you use to indicate a particular page in a tblspace	Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier. Value must be an unsigned integer between 0 and 16,777,215, inclusive.

Element	Purpose	Key Considerations
owner	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; see *HCL OneDB™ Guide to SQL: Syntax*.
rowid	Identifies the rowid of the row whose contents you want to display. The rowid is displayed as part of **oncheck -pD** output	Value must be an unsigned integer between 0 and 4,277,659,295, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
table	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; see *HCL OneDB™ Guide to SQL: Syntax*.
tblspacenum	Identifies the tblspace whose contents you want to display	Value must be an unsigned integer between 0 and 208,666,624, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.

If you specify an internal rowid (expressed as a hexadecimal value), the rowid maps to a particular page, and all rows from that page are printed.

If you specify a logical page number (expressed as a decimal), all the rows of the tblspace number with the logical page number are printed.

If you specify a fragment, all the rows in the fragment are printed, with their rowids, forward pointers, and page type.

If you specify a table, all the rows in the table are printed, with their rowids, forward pointers, and page type.

If you specify a database, all the rows in all the tables in the database are printed. TEXT and BYTE column descriptors stored in the data row are printed, but TEXT and BYTE data itself is not.

The **-pD** option prints the same information as **-pd**. In addition, **-pD** prints TEXT and BYTE values stored in the tblspace or header information for simple large objects stored in a blobspace blobpage. The following example show different options for the **oncheck -pd** and **oncheck -pD** commands:

```
oncheck -pd stores_demo:customer,frgmnt1
oncheck -pd stores_demo:customer
oncheck -pD stores_demo:customer 0x101
```

The following example shows a partial output of an **oncheck -pD** command:

```
oncheck -pD multipart:t1 :

TBLspace data check for multipart:informix.t1
```

```
                  Table fragment partition part_1 in DBspace dbs1
page_type  rowid    length fwd_ptr
HOME       101      24     0
    0:  0  0  0  a 47 48 49 20 20 20 20 20 20 20 20 20   ....GHI
   16: 20 20 20 20 20 20 20 20
                                         ........
```

oncheck -pk, -pK, -pl, -pL: Display index information

```
>>-oncheck-----------------------------------------------------> 

>----+- -pk-+--+---+--database--+---------------------------------------+---><
     +- -pK-+  '-x-'            '-:--+--------+--table--+-------------+-'
     +- -pl-+                        '-owner.-'         '-#index_name-'
     '- -pL-'
```

The **-pk** option performs the same checks as the **-ci** option and in addition, displays the key values for all indexes on the specified table as it checks them.

The **-pK** option performs the same checks as the **-cI** option and in addition, displays the key values and rowids as it checks them.

The **-pl** option performs the same checks as the **-ci** option and displays the key values, but checks only leaf-node index pages. It ignores the root and branch-node pages.

The **-pL** option performs the same checks as the **-cI** option and displays the key values and rowids, but checks only leaf-node index pages. It ignores the root and branch-node pages.

Element	Purpose	Key Considerations
database	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
index_name	Specifies the name of the index that you want to check for consistency	Index must exist on table and in database specified. Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
owner	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; see *HCL OneDB™ Guide to SQL: Syntax*.
table	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; see *HCL OneDB™ Guide to SQL: Syntax*.
-x	Places a shared lock on the table when you check and print an index	For complete information, see Turn On Locking with -x on page 342.

If any of the **oncheck** options detect inconsistencies, you are prompted for confirmation to repair the problem index. If you specify the **-y** (yes) option, indexes are automatically repaired. If you specify the **-n** (no) option, the problem is reported but not repaired; no prompting occurs.

The following example displays information about all indexes on the **customer** table:

```
oncheck -pl -n stores_demo:customer
```

The following example displays information about the index **zip_ix**, which was created on the **customer** table:

```
oncheck -pl -n stores_demo:customer#zip_ix
```

By default, the database server does not place a shared lock on the table when you check an index with the **oncheck -pk, -pK, -pl,** or **-pL** options unless the table uses page locking. For absolute assurance of a complete index check, you can execute **oncheck -pk**, **oncheck -pK** , **oncheck -pl** , or **oncheck -pL** with the **-x** option. With the **-x** option, **oncheck** places a shared lock on the table, and no other users can perform updates, inserts, or deletes until the check has completed. For more information on using the **-x** option, Turn On Locking with -x on page 342.

For more information on **oncheck -ci**, see oncheck -ci and -cI: Check index node links on page 329. For more information index pages, see Structure of B-Tree Index Pages on page 299.

oncheck -pp and -pP: Display the contents of a logical page

```
>>-oncheck----------------------+----------------------+------->
                                '- -pw-+------------+--'
                                       '--filename--'
>--+- -pp--+-database--:--+--------+--table--+--------------+--rowid-+-+-><
   |       |              '-owner.-'         +-,--frag_dbs--+        | |
   |       |                                 '-%--frag_part-'        | |
   |       '-tblspacenum--logical pagenum----------------------------' |
   '- -pP--chunknum--logical pagenum-----------------------------------'
```

Element	Purpose	Key Considerations
database	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax.*
chunknum	Specifies a decimal value that you use to indicate a particular chunk	Value must be an unsigned integer greater than 0. Chunk must exist.
frag_dbs	Specifies the name of a dbspace that contains a fragment you want to check for consistency	Dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
frag_part	Specifies the partition name of the fragment to be checked. This is useful in cases where more than one fragment	For fragmented tables or an index that use expression-based or round-robin distribution schemes, you can create multiple partitions, which are collections

Element	Purpose	Key Considerations
	of a table was created in the same dbspace.	of pages for a table or index, within a single dbspace. This partition is referred to as a *fragment partition* or *fragpart*.
logical pagenum	Specifies an integer value that you use to indicate a particular page in a tblspace	Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier. Value must be an unsigned integer between 0 and 16,777,215, inclusive.
owner	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; see *HCL OneDB™ Guide to SQL: Syntax*.
rowid	Identifies the rowid of the row whose contents you want to display. The rowid is displayed as part of **oncheck -pD** output	Value must be an unsigned integer between 0 and 4,277,659,295, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.
table	Specifies the name of the table that you want to check for consistency	Table exists when you execute the utility. Syntax must conform to the Table Name segment; see *HCL OneDB™ Guide to SQL: Syntax*.
tblspacenum	Identifies the tblspace whose contents you want to display	Value must be an unsigned integer between 0 and 208,666,624, inclusive. Value can be expressed as an unsigned integer or hexadecimal that begins with 0x identifier.

The **-pp** option has the following syntax variations:

Invocation	Explanation
oncheck -pp tblspc lpn <pages>	Displays the contents of a logical page using a tblspace number and logical page number. You can also specify an optional parameter specifying the number of pages to be printed.
oncheck -pp *tblspc lpn -h*	Displays only the header of a logical page using a tblspace number and logical page number.
oncheck -pp *database:table rowid*	Displays the contents of a logical page using a database name, table name, and the HCL OneDB™ internal rowid. You can obtain this internal rowid with the **oncheck -pD** command. This internal rowid is not the serial rowid that is assigned in tables created with the CREATE TABLE tabname

Invocation	Explanation
	WITH ROWIDS statement. For more information, see Definition of Rowid on page 296

The page contents appear in ASCII format. The display also includes the number of slot-table entries on the page. The following example shows different invocations of the **oncheck -pp** command:

```
oncheck -pp stores_demo:orders 0x211  # database:owner.table,   # fragment rowid
oncheck -pp stores_demo:informix.customer,frag_dbspce1 0x211
oncheck -pp 0x100000a 25 # specify the tblspace number and # logical page number
```

The **-pP** option provides the following syntax variations:

Invocation	Explanation
oncheck -pP chunk# offset pages	Displays the contents of a logical page using a chunk number and an offset. You can also specify an optional parameter specifying the number of pages to be printed.
oncheck -pP *chunk# offset -h*	Displays only the header of a logical page using a chunk number and an offset.

Note: The output for chunk page displays both the `start` and the `length` fields in decimal format.

Note:

The -pw option is required only when the Storage space encryption feature is enabled and no stash file is in use. Supply an optional path to a file containing the keystore password, otherwise oncheck will prompt for a password before displaying the requested page(s).

The following example shows typical output using the **onstat -pP** command:

```
oncheck -pP 1 5 2
addr          stamp          nslots          flag       type          frptr        frcnt        next        prev
stamp          100005          250181          2          1000        ROOTRSV          320        1716        0
0       250181        slot        ptr        len        flg
...
addr       stamp          nslots          flag       type          frptr        frcnt        next        prev
stamp       100005    6       250182       2       1000       ROOTRSV       128       1908       0       0
250182       slot       ptr       len       flg       1       24       56       0
2       80       48         0
```

oncheck -pr and pR: Display reserved-page information

The -pr option performs the same checks as oncheck -cr command and displays the reserved-page information.

```
>>-oncheck----+- -pr-+---+----------------------+------------><
              '- -pR-'   '- -pw-+------------+--'
                                '--filename--'
```

The -pR option performs the same checks as the oncheck -cR command, displays the reserved-page information, and displays detailed information about logical-log and physical-log pages, including marking the start and end of the active physical-log pages.

The following example show output of the oncheck -pr command:

```
Validating IBM Informix Dynamic Server reserved pages

    Validating PAGE_PZERO...

    Identity                       IBM Informix Dynamic Ser
                                   ver Copyright 2001, 2016
                                     IBM Corporation
    Database system state          0
    Database system flags          0xc039
        64-bit server
        BigChunk page flags are not in use
        Encryption-at-rest is enabled using cipher 'aes192'
        The ROOT Dbspace is encrypted
    Page Size                      2048 (b)
    Date/Time created              05/12/2016 18:01:09
    Version number of creator      28
    UID of rootdbs creator         200
    Index Page Logging             OFF
    HA Disk Owner                  <null>
```

```
Validating IBM Informix Dynamic Server reserved pages

    Validating PAGE_PZERO...

    Identity                       IBM Informix Dynamic Ser
                                   ver Copyright 2001, 2016
                                     IBM Corporation
    Database system state          0
    Database system flags          0xc039
        64-bit server
        BigChunk page flags are not in use
    Page Size                      2048 (b)
    Date/Time created              05/12/2016 18:01:09
    Version number of creator      28
    UID of rootdbs creator         200
    Index Page Logging             OFF
    HA Disk Owner                  <null>
```

If you have changed the value of a configuration parameter, but you have not yet reinitialized shared memory, the oncheck -pr and oncheck -pR commands detect the inconsistency and return an error message.

Note:

The -pw option is required only when the Storage space encryption feature is enabled and no stash file is in use. Supply an optional path to a file containing the keystore password, otherwise oncheck will prompt for a password before displaying the requested page(s).

If you have changed the value of a configuration parameter, but you have not yet reinitialized shared memory, oncheck -pr and oncheck -pR detect the inconsistency and return an error message.

oncheck -pt and -pT: Display tblspaces for a Table or Fragment

The oncheck -pt and oncheck -pT options print a tblspace report for a specific table or fragment. The only difference between these options is that oncheck -pT prints more information, including some index-specific information.

```
>>-oncheck------------------------------------------------------>

>----+- -pt-+--database--+-------------------------------------+---><
     '- -pT-'            '-:--+--------+--table--+-----------+-'
                              '-owner.-'         '-,frag_dbs-'
```

Table 117. Options of the oncheck -pt and oncheck -pT commands

Element	Purpose	Key Considerations
database	Specifies the name of a database that you want to check for consistency	Syntax must conform to the Identifier segment; see Identifier on page .
frag_dbs	Specifies the name of a dbspace that contains a fragment you want to check for consistency	The dbspace must exist and contain the fragment that you want to check for consistency. Syntax must conform to the Identifier segment; see Identifier on page .
owner	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Owner Name segment; see Owner name on page .
table	Specifies the name of the table that you want to check for consistency	The table must exist. Syntax must conform to the Identifier segment; see Identifier on page .

The -pt option prints a tblspace report for the table or fragment with the specified name and database. If you do not specify a table, the option displays this information for all tables in the database. The report contains general allocation information, including the maximum row size, the number of keys, the number of extents, their sizes, the pages allocated and used per extent, the current serial value, and the date that the table was created. The -pt output prints the page size of the tblspace, the number of pages (allocated, used, and data) in terms of logical pages.

The **TBLspace Flags** field shows information about the tblspace configuration, including whether the tblspace is used for Enterprise Replication or time series data.

The **Extents** fields list the physical address for the tblspace **tblspace** entry for the table and the address of the first page of the first extent. The extent list shows the number of logical and physical pages in every extent.

The -pT option prints the same information as the -pt option. In addition, the -pT option displays:

- Index-specific information
- Page-allocation information by page type (for dbspaces)
- The number of any compressed rows in a table or table fragment and the percentage of table or table-fragment rows that are compressed

  If table or fragment rows are not compressed, the "Compressed Data Summary" section does not appear in the output.

Plan when you want to run the -pT option, because it forces a complete scan of partitions.

Output for both -pt and -pT contains listings for **Number of pages used**. The value shown in the output for this field is never decremented because the disk space allocated to a tblspace as part of an extent remains dedicated to that extent even after you free space by deleting rows. For an accurate count of the number of pages currently used, see the detailed information about tblspace use (organized by page type) that the -pT option provides.

Example of oncheck -pt Output

The following example shows output of the oncheck -pt command:

```
TBLspace Report for testdb:tab1

Physical Address               2:10
Creation date                  10/07/2004 17:01:16
TBLspace Flags                 801        Page Locking
                                              TBLspace use 4 bit bit-maps
Maximum row size               14
Number of special columns      0
Number of keys                 0
Number of extents              1
Current serial value           1
Pagesize (k)                   4
First extent size              4
Next extent size               4
Number of pages allocated      340
Number of pages used           337
Number of data pages           336
Number of rows                 75806
Partition partnum              2097154
Partition lockid               2097154

Extents
       Logical Page     Physical Page        Size Physical Pages
                 0             2:106         340        680
```

Example of oncheck -pT Output

The following example shows output of the oncheck -pT command:

```
TBLspace Report for database_a:nilesh.table_1a

                  Table fragment partition dbspace1 in DBspace dbspace1

    Physical Address               3:5
    Creation date                  03/21/2009 15:35:47
    TBLspace Flags                 8000901    Page Locking
                                              TBLspace contains VARCHARS
                                              TBLspace use 4 bit bit-maps
                                              TBLspace is compressed
    Maximum row size               80
    Number of special columns      1
    Number of keys                 0
    Number of extents              1
    Current serial value           100001
    Current SERIAL8 value          1
    Current BIGSERIAL value        1
    Current REFID value            1
    Pagesize (k)                   2
    First extent size              8
    Next extent size               8
    Number of pages allocated      24
    Number of pages used           22
    Number of data pages           14
    Number of rows                 500
    Partition partnum              3145730
    Partition lockid               3145730

    Extents
         Logical Page     Physical Page        Size Physical Pages
                    0          3:16053          24         24

Type                  Pages      Empty  Semi-Full       Full  Very-Full
    ---------------- ---------- ---------- ---------- ---------- ----------
    Free                      9
    Bit-Map                   1
    Index                     0
    Data (Home)              14
    Data (Remainder)          0          0          0          0          0
                     ----------
    Total Pages              24

    Unused Space Summary

        Unused data bytes in Home pages                1177
        Unused data bytes in Remainder pages              0

    Home Data Page Version Summary

                 Version                               Count

                       0 (current)                       14
```

```
    Compressed Data Summary

    Number of compressed rows and percentage of compressed rows  500 100.00
```

Note: oncheck -p[tT] now indicates the last time each index fragment was used for a query. This access time is stored on the partition page on disk, it will survive an instance restart.

Turn On Locking with -x

The **-x** option can be appended to the **-ci**, **-cI**, **-pk**, **-pK**, **-pl**, and **-pL** options to place a shared lock on affected tables. While the table is locked, no other users can perform inserts, updates, and deletions while **oncheck** checks or prints the index. Without the **-x** option for tables with row locking, **oncheck** only places an IS (intent shared) lock on the table, which prevents actions such as dropping the table or the indexes during the check.

For example, the following sample command instructs **oncheck** to lock indexes for the **customer** table while it validates the order of key values, validates horizontal links, and ensures that no node appears twice in the index:

```
oncheck -cix stores_demo:customer
```

When you specify option **-x**, **oncheck** locks indexes for tables that use row locking. If **oncheck** detects page-lock mode, it displays a warning message and places a shared lock on the table regardless.

Send Special Arguments to the Access Method with -u

You can use the **-u** option to send special arguments to the access method. The possible arguments depend on the access method. For example, the R-tree access method supports the **display** option, as the following example shows:

```
oncheck -pl -u "display"
```

Use commas to separate multiple arguments in the argument string.

For information on valid arguments for your access method, refer to the user manual for your access method.

Return Codes on Exit

The **oncheck** utility returns the following codes on exit.

```
GLS failures:-1
Invalid srial/key:2
Onconfig access error:2
Invalid onconfig settings:2
Invalid arguments to oncheck:2
Error connecting database server:1
Warning reported by oncheck:1
error detected by oncheck:2
no errors detected by oncheck:0
```

Windows™ only:

```
Not properly installed:1
Authentication error:2
```

The onclean utility

Use the onclean utility to force a shut down of the database server when normal shut down with the onmode utility fails or when you cannot restart the server. The onclean utility attempts to clean up shared memory, semaphores, and stops database server virtual processes.

On UNIX™ and Linux™, you must be user **root** or **informix** to run the onclean command. On Windows™, you must be in the **Informix-Admin** group to run the command.

```
onclean {[ < -FILE option > (explicit id ) ]|[ -k][ -y]|[{ -V | -version}]}
```

Table 118. Syntax Elements of the onclean Command

Element	Purpose
-k	Shuts down a server that is online by stopping database server virtual processes and attempting to clean up the remaining semaphores and shared-memory segments, even if they are still running.
-V	Displays short version information.
-version	Displays full version information.
-y	Does not prompt for confirmation.

Usage

Use the onclean utility to stop the database server only if the onmode utility is unable to shut it down or you cannot restart the server. Perhaps the database server shut down in an uncontrolled way and cannot recover, or it is hung. If the database server fails to restart, the previous instance of the database server is still attached to the shared-memory segments. Check the message log to see if the database server shut down abnormally. The onclean utility stops all oninit processes and attempts to remove all shared-memory segments and semaphores that are recorded in the **$ONEDB_HOME/etc/.conf.$ONEDB_SERVER** file.

Attention: Use the onclean utility with caution. When you run onclean, any pending transactions and processes fail to complete, and user sessions are disconnected abruptly. However, the database server rolls back transactions when it restarts.

The **ONEDB_HOME**, **ONEDB_SERVER**, **ONEDB_ SQLHOSTS**, and **ONCONFIG** environment variables must be set with valid values to run this utility.

The onclean command that you use depends on the situation:

- If you are not sure whether the database server is offline, use the onclean command without options. If the database server is still online, a message appears directing you to run the onclean -k command.
- If the database server is offline, use the onclean command.
- If the database server is online and you are sure that you want to force it to shut down, use the onclean -k command.

You can use the onclean utility only to shut down the local database server; you cannot use it to shut down a remote database server. The onclean utility should not be used to shut down an entire high-availability cluster or a remote database server.

The onclean utility might not be able to clean up shared memory segments that were in use by the database server in every situation. The onclean utility attempts to terminate only oninit processes. The onclean utility does not succeed in the following situations:

- If a non-database server process is attached to the shared memory segment before running the onclean command, the onclean utility does not stop this process to remove the shared memory segment.
- The onclean might not be able to guarantee a clean server startup is when an application or database server utility is connected to a network port. If the user tries to initialize a database server instance on the same network port, then the database server cannot start the listener thread and fails to start. The onclean utility does not stop the application to free the network port.

You can automate shutting down the database server with the onshutdown script, which calls the onclean -ky command if necessary.

Return Codes

0

Successful

1

Failure because of one of the following problems:

- Incorrect environment variable settings
- Incorrect privileges to run the onclean command
- Incorrect command syntax
- Corrupted information
- Running the onclean command without the -k option on a server that is still online

2

Failure because one or more OS system calls used by onclean returned an error.

The onshutdown script

Use the onshutdown script to automate shutting down the database server. The script attempts to shut down the server normally. If the server has not shut down after a specified time, the script forces the server to shut down.

The onshutdown script first runs the onmode -ky command. After a specified wait time, the script runs the onclean -ky command.

On UNIX™ and Linux™, you must be user **root** or **informix** to run the onshutdown script. On Windows™, you must be in the **Informix-Admin** group to run the onshutdown script.

```
                   (1)
>>-+-onshutdown.sh------+--+---------+------------------------><
   |               (2)  |  '-timeout-'
   '-onshutdown.bat-----'

Notes:
```

Table 119. Syntax Elements of the onshutdown Script

Element	Purpose
timeout	The number of seconds after the onmode -ky command has been run before running the onclean -ky command. Must be a positive integer from 10 to 60. The default value is 30 seconds.

Usage

Use the onshutdown script only when forcing the database server to shut down would be appropriate.

Attention: Use the onshutdown script with caution. If the script needs to run the onclean -ky command, any pending transactions and processes fail to complete, and user sessions are disconnected abruptly. However, the database server rolls back transactions when it restarts.

The **ONEDB_HOME**, **ONEDB_SERVER**, **ONEDB_ SQLHOSTS**, and **ONCONFIG** environment variables must be set with valid values to run this utility.

You can only use the onshutdown script to shut down the local database server; you cannot use it to shut down a remote database server. The onshutdown script should not be used to shut down an entire high-availability cluster or a remote database server instance.

The onshutdown script has a 10 second time period during which it can be aborted.

The oncmsm utility

Use the oncmsm utility to start or shut down a Connection Manager, load a new configuration file into a Connection Manager to modify the Connection Manager's settings, or update the format of a configuration file.

Syntax

UNIX syntax diagram:

oncmsm { [**-c** *configuration_file*] [**-n** *new_configuration_file*] | { **-r** | **-k** } *connection_manager_name* }

Windows syntax diagram:

oncmsm { { **-i** **-c** *configuration_file* | [**-c** *configuration_file*] **-n** *new_configuration_file* | { **-r** | **-k** | **-u** } *connection_manager_name* | *connection_manager_name* } | }

Element	Purpose	Key considerations
-c	Starts the Connection Manager or converts a configuration file to the current Connection Manager format.	
connection_manager_name	Specifies the name of a Connection Manager instance.	
-i	Installs the Connection Manager as a Windows™ service.	This option is valid for Windows™ platforms only.
-k	Shuts down a specific instance of the Connection Manager.	
-n	Specifies the name of a converted configuration file.	
new_configuration_file	The name of file that is output to the `$ONEDB_HOME/etc` directory as part of the format-conversion process.	
configuration_file	The name of the configuration file located in the `$ONEDB_HOME/etc` directory.	If the configuration file is not specified, the Connection Manager attempts to load `$ONEDB_HOME/etc/cmsm.cfg`.
-r	Reloads the Connection Manager settings without stopping and restarting the Connection Manager.	
-u	Uninstall the Connection Manager Windows™ service.	This option is valid for Windows™ platforms only.

Usage

Run the oncmsm utility from the command line to initialize the Connection Manager. You can add, change, or delete Service Level Agreements (SLAs) while the Connection Manager is running and then reload the configuration file.

The Connection Manager configuration file in versions of HCL OneDB™ Client Software Development Kit (Client SDK) prior to version 3.70.xC3 are incompatible with the current version of the Connection Manager. You must convert configuration files from versions prior to 3.70.xC3. You must have read permission on the configuration file you want to convert and write permission on the configuration file you want to create.

UNIX Only: The following users can run the oncmsm utility:

- User **informix**
- User **root**, if the user has privileges to connect to the **sysadmin** database
- A member of the **DBSA** group, if the user has privileges to connect to the **sysadmin** database

Windows Only: The following users can run the oncmsm utility:

- A member of the **Informix-Admin** group
- User **administrator**, if the user has privileges to connect to the **sysadmin** database
- A member of the **DBSA** group, if the user has privileges to connect to the **sysadmin** database

You must install the oncmsm utility as a service before you can start it.

The oncmsm utility can be started two ways:

- Run an oncmsm command.
- Click **Start > Control Panel > Administrative Tools > Services** and then start oncmsm.

If you are using multiple Connection Managers, you can run onstat -g cmsm to display the names of Connection Manager instances.

Example

Example 1: Starting a Connection Manager (UNIX™)

For the following example the Connection Manager's `configuration_file_1` exists in the `$ONEDB_HOME/etc` directory. To start the Connection Manager, run the following command on the computer that the Connection Manager is installed on:

```
oncmsm -c configuration_file_1
```

The Connection Manager starts.

Example

Example 2: Starting a Connection Manager (Windows™)

For the following example the Connection Manager's `configuration_file_1` exists in the `$ONEDB_HOME/etc` directory. To start the Connection Manager, run the following commands on the computer that the Connection Manager is installed on:

```
oncmsm -i -c configuration_file_2
oncmsm connection_manager_2
```

The Connection Manager named **connection_manager_2** starts.

Example

Example 3: Stopping a Connection Manager

To stop the Connection Manager, run the following command on the computer that the Connection Manager is installed on:

```
oncmsm -k connection_manager_3
```

The Connection Manager named **connection_manager_3** stops.

Example

Example 4: Reloading Connection Manager settings

For the following example, `$ONEDB_HOME\etc\configuration_file_4` for a Connection Manager named **connection_manager_4** has changed. To update the Connection Manager's settings, run the following command on the computer that **connection_manager_4** is installed on:

```
oncmsm -r connection_manager_4
```

Example

Example 5: Converting a Connection Manager configuration file to a current format

For the following example the Connection Manager's configuration file that is named `cmsm.cfg` exists in the `$ONEDB_HOME/etc` directory. To start the Connection Manager, run the following command on the computer that the Connection Manager is installed on:

```
oncmsm -n configuration_file_5
```

The oncmsm utility converts `cmsm.cfg` to the current configuration file format, and then outputs a file named `configuration_file_5` into `$ONEDB_HOME/etc/`.

Example

Example 6: Converting a specific Connection Manager configuration file to a current format

For the following example the Connection Manager's configuration file that is named `configuration_file_4` exists in the `$ONEDB_HOME/etc` directory. To start the Connection Manager, run the following command on the computer that the Connection Manager is installed on:

```
oncmsm -c configuration_file_6 -n configuration_file_7
```

The oncmsm utility converts `configuration_file_6` to the current configuration file format and then outputs a file named `configuration_file_7` into `$ONEDB_HOME/etc/`.

Example

Example 7: Uninstalling a Connection Manager (Windows™)

For the following example, you have installed a Connection Manager named **connection_manager_4** as a Windows™ service. To unistall the Connection Manager, run the following command on the computer that the Connection Manager is installed on:

```
oncmsm -u connection_manager_4
```

The oncmsm utility uninstalls the Connection Manager.

The onconfig_diff utility

Use the onconfig_diff utility to compare two `onconfig` files.

Syntax

```
>>-onconfig_diff--+- -d---------------------------------+----><
                  '- -c-- -f--filepath_1-- -s--filepath_2-'
```

Element	Description
-d	Compares the current `onconfig` settings to the default settings.
-c	Compares one `onconfig` file to another.
-f *filepath_1*	Specifies the first file name to compare. Provide the path to the file unless the file is in the `$ONEDB_HOME/bin` directory.
-s *filepath_2*	Specifies the second file name to compare. Provide the path to the file unless the file is in the `$ONEDB_HOME/bin` directory.

Usage

Run the onconfig_diff utility to compare two different `onconfig` files. The onconfig_diff utility is in `$ONEDB_HOME/bin`.

The two files that you want to compare must be in the same directory.

Here are some ways that you can use the utility:

- Compare your current `onconfig` with the `onconfig.std` of same version.
- Compare your current `onconfig` with the `onconfig.std` of a newer version.
- Compare two `onconfig` files from different servers.

Example

Example

In this example, the `onconfig.std` file is compared against the `onconfig.production` file:

```
$ onconfig_diff -c -f onconfig.std -s onconfig.production
```

Here is the output from this command:

```
==========================================
File 1: onconfig.std
File 2: onconfig.production
==========================================
Parameters Found in File 1, not in File 2
==========================================

FULL_DISK_INIT  0
```

```
NETTYPE         ipcshm,1,50,CPU

NUMFDSERVERS    4
...
==============================================
Parameters Found in File 2, not in File 1
==============================================

JVPJAVAHOME    $ONEDB_HOME/extend/krakatoa/jre

...
==============================================
Parameters Found in both files, but different
==============================================
ROOTPATH

File 1: $ONEDB_HOME/tmp/demo_on.rootdbs
File 2: /usr2/support/grantf/g1150fc8/rootdbs

LOGFILES

File 1: 6
File 2: 10

LOGSIZE

File 1: 10000
File 2: 3000
...
```

The ondblog utility

Use the ondblog utility to change the logging mode for one or more databases.

```
>>-ondblog--+-buf-------+--+-------------+---------------------><
            +-unbuf-----+  | .-,-------. |
            +-nolog-----+  | V         | |
            +-ansi------+  +---db_list-+-+
            +-cancel----+  '- -f--dbfile-'
            +- -V-------+
            '- -version-'
```

Element	Purpose	Key Considerations
buf	Sets the logging mode so that transaction information is written to a buffer before it is written to a logical log	None.
unbuf	Sets the logging mode so that data is not written to a buffer before it is written to a logical log	None.
nolog	Sets the logging mode so that no database transactions are logged	None.

Element	Purpose	Key Considerations
ansi	Changes database logging to be ANSI compliant	Once you create or convert a database to ANSI mode, you cannot change it back to any of the other logging modes.
cancel	Cancels the logging-mode change request before the next level-0 backup occurs	None.
-f *dbfile*	Changes the logging status of the databases that are listed (one per line) in the text file whose pathname is given by *dbfile*	This command is useful if the list of databases is long or used often.
db_list	Names a space-delimited list of databases whose logging status is to be changed	If you do not specify anything, all databases that the database server manages are modified.

Usage

If you turn on transaction logging for a database, you must create a level-0 backup of all of the storage spaces that contain data in the database before the change takes effect.

For more information and examples of logging modes, see Modify the database-logging mode with ondblog on page .

Alternatively, you can change the logging mode by using an SQL administration API command with the **alter logmode** argument.

You cannot use the ondblog utility on High-Availability Data Replication (HDR) secondary servers, remote standalone (RS) secondary servers, or shared disk (SD) secondary servers.

Return codes

The ondblog utility logs messages in the BAR_ACT_LOG file.

For many of the return codes, you can check the ON-Bar logs to find the source of the problem:

1. Check the BAR_ACT_LOG file for accompanying messages.
2. Set the BAR_DEBUG configuration parameter to a positive integer and retry the operation.
3. Check the ON-Bar debug log file.

Table 120. Return codes for the ondblog utility

Return code	Description	User action
1	An error reading the `onconfig` file.	Check that the `onconfig` file is in the `$ONEDB_HOME/etc/$ONCONFIG` directory. If the BAR_ACT_LOG and BAR_DEBUG_LOG configuration

Table 120. Return codes for the ondblog utility (continued)

Return code	Description	User action
		parameters are set, make sure that the files are valid.
2	A linked list error.	See the accompanying message.
3	The user is not authorized to run the command.	Run the ondblog commands as the **root** user, user **informix**, as a Windows administrator, or as the owner of the database server.
4	Failed to set the **ONEDB_ SHMBASE** environment variable to -1.	Contact HCL Software Support.
5	The database server is not online.	Start the database server.
6	The command option is invalid.	Correct the spelling of the option.
7	Failed to communicate with the backup utility.	Check that the database server is online and that ON-Bar is configured.
9	Failed to allocate memory.	Check the ON-Bar logs for more information. You might need to ask your System Administrator to either increase your swap space or to install more memory in your system.
16	Failed to open the file.	Check the ON-Bar logs for more information.
17	Cannot change to the specified logging mode.	Check the ON-Bar logs for more information.
18	Failed to change the logging mode.	Check the ON-Bar logs for more information.
19	An SQL error occurred.	Check the ON-Bar logs for more information.
20	An empty list problem occurred.	Check the ON-Bar logs for more information.

The oninit utility

The oninit utility starts the database server.

On UNIX™, Linux™, you must be logged in as user **root**, user **informix**, or the non-root database server owner to run the oninit utility. User **informix** should be the only member of the group **informix**. Run the oninit command from the command line.

You can allow users who belong to the DBSA group to run the oninit command. See Allow DBSA group users to run the oninit command (UNIX) on page 356.

On Windows™, HCL OneDB™ runs as a Windows™ service. Any user who has appropriate permissions to start a Windows™ service is able to start the HCL OneDB™ service. The **Services** control application runs the oninit utility with any options that you supply.

Syntax

```
>>-oninit--+-------------------------+-------------------------->
           |                     (1) |
           '-|  -FILE option   |-----'

>--+-------------------------------------------+---------------><
   +-| Other options for starting the server |-+
   +-| Initialize disk space |-----------------+
   '- -PHY-------------------------------------'

Other options for starting the server

|--+-+-----+--+--------------------------+-+--+-------------+--->
   | '- -j-'   |      .-600-------------. | |  +- -SDS=alias-+
   |          '- -w--+-+-------------+-+-' |  '- -D---------'
   |                   '-max_seconds-'     |
   '-+- -s-+-------------------------------'
     '- -S-'

>--+-----------------------+--+-----+--+-----+--+-----+---------|
   |        .-,--------.   |  '- -p-'  '- -y-'  '- -v-'
   |        V          |   |
   '- -U--+---username-+-+-'
          '-" "----------'

Initialize disk space

|-- -i--+-----+--+-----+--+-----+------------------------------->
        +- -j-+  '- -y-'  '- -v-'
        '- -s-'

>--+--------------------------+--+-----------------------+------|
   |      .-600-------------. |  |        .-,--------.   |
   '- -w--+-+-------------+-+-'  |        V          |   |
            '-max_seconds-'      '- -U--+---username-+-+-'
                                        '-" "----------'
```

Table 121. oninit command elements

Element	Purpose	Key Considerations
-D	Starts the database server with Enterprise Replication and high-availability cluster replication disabled.	

Table 121. oninit command elements (continued)

<table>
<tr><th>Element</th><th>Purpose</th><th>Key Considerations</th></tr>
<tr><td>-i</td><td>Initializes disk space for the root dbspace so that it can be used by the database server and starts the database server.</td><td>Disk space needs to be initialized only once to prepare data storage for the server.

By default, to prevent data loss, you cannot reinitialize disk space. To reinitialize disk space for an existing root dbspace, you must set the FULL_DISK_INIT configuration parameter to <code>1</code> and then run the oninit -i command.

See Initialize disk space for the root dbspace on page 356.</td></tr>
<tr><td>-j</td><td>Starts the server in administration mode.</td><td>See Start the server in administration mode on page 355.</td></tr>
<tr><td>-p</td><td>Starts the database server without deleting temporary tables.</td><td>If you use this option, the database server starts more rapidly, but space used by temporary tables left on disk is not reclaimed.</td></tr>
<tr><td>-PHY</td><td>Starts the server as of most current checkpoint. The -PHY option is used to tell the server to do only physical recovery without logical recovery.</td><td>This option is normally used to start a secondary server. You must run one of the following commands to connect the secondary server to the primary server:
<pre>onmode -d secondary
onmode -d RSS</pre>The connection of the secondary server to the primary server fails if the most recent checkpoint on the primary server was not performed on the secondary server.</td></tr>
<tr><td>-s</td><td>Starts the server in quiescent mode.</td><td>The database server must be shut down when you use this option.

When the database server is in quiescent mode, only the user informix can access the database server.</td></tr>
<tr><td>-S</td><td>Starts database server in quiescent mode as a standard server with high-availability data replication disabled.</td><td>When the database server is in quiescent mode, only the user informix can access the database server.</td></tr>
<tr><td>-U username</td><td>Specifies which users can access the server in administration mode for the current session.</td><td>The informix user and members of the DBSA group are always administration mode users.

See Start the server in administration mode on page 355.</td></tr>
</table>

Table 121. oninit command elements (continued)

Element	Purpose	Key Considerations
-v	Displays verbose informational messages while the server is starting.	
-w *max_seconds*	Starts the database server and waits to indicate success or failure until the server is completely started in online mode or the number of seconds specified by *max_seconds* elapses.	The default number of seconds to wait is 600. This option is not valid on secondary servers in a high-availability cluster. See Start the server with a script on page 356.
-y	Prevents verification prompts.	The -y option automatically answers yes to all the verification prompts.

Usage

By default, the oninit utility shows verification prompts during server startup. You can suppress verification prompts by including the -y option. You can view verbose informational messages by including the -v option. On UNIX™, Linux™, oninit output is shown to standard output. On Windows™, you can view oninit output by setting the **ONINIT_STOUT** environment variable to save the output to a file.

You can start the server in different operating modes. By default, if you run the oninit command without options, the server starts in online mode. When the database server is in online mode, all authorized users can access the server.

If you run an oninit -FILE command, you do not need to set local environment variables before you start the database server. The database server automatically uses the environment variables that are set as values in the onconfig file.

Start the server in administration mode

Administration mode is an administrator-only mode you can use to perform maintenance operations including those that require running SQL or DDL commands. When in administration mode, the database server only accepts connection requests from the following users:

- The **informix** user
- Members of the DBSA group
- Users specified by the oninit -U command or the onmode -j -U command, for the current session. The -U option overrides any users listed by the ADMIN_MODE_USERS configuration parameter in the `onconfig` file.
- Users specified by the ADMIN_MODE_USERS configuration parameter

Use the -U option with a list of comma-separated user names to add administration mode users, such as: `Karin,Sarah,Andrew`.

Use the -U " " option to remove all administration mode users except the **informix** user and members of the DBSA group: oninit -U " ".

Initialize disk space for the root dbspace

The first time you install HCL OneDB™ on your system, disk space for the root dbspace for the database server needs to be initialized. The root dbspace is specified by the ROOTPATH configuration parameter.

If you performed a typical installation and chose to create a database server or you performed a customer installation, disk space was automatically initialized. Otherwise, you must initialize disk space by running the oninit -i command.

If the DISK_ENCRYPTION configuration parameter is set when you initialize the root dbspace, the root dbspace is encrypted.

If necessary, you can reinitialize disk space. Reinitializing disk space destroys all existing data managed by the database server. The database server must be offline when you reinitialize.

By default, you cannot reinitialize a root dbspace that is being used by the database server. Disk initialization fails if a page zero exists at the root path location (at the first page of the first chunk location). You can allow disk reinitialization of an existing root dbspace by setting the FULL_DISK_INIT configuration parameter to `1`.

Start the server with a script

You can use the oninit -w command in customized startup scripts and to automate startup. The -w option forces the server to wait until startup is completely successfully before indicating that the server is in online mode by returning to the shell prompt with a return code of `0`. If the server is not in online mode within the timeout period, the server returns a return code of `1` to the shell prompt and writes a warning message in the online log.

The default timeout is 600 seconds (10 minutes), which you can modify to any integer value.

After running the following command, if the server fails to start within 60 seconds, a code of `1` is returned to the prompt:

```
oninit -w 60
```

To determine the reason for the server failing to start, check the online log. You might need to increase the timeout value. When you use the oninit -w command in a script, you can check whether the server is online with the onstat - (Print output header) command.

Allow DBSA group users to run the oninit command (UNIX™)

To allow users who belong to the DBSA group, other than the user **informix**, to run the oninit command, log in as the user **root** and change the permissions on the oninit utility in the `$ONEDB_HOME/bin` directory from 6754 to 6755.

The -FILE option

On UNIX™, you can use the -FILE option to run certain HCL OneDB™ utilities with the local environment variables that you set in your `onconfig` file. You do not have to set local environment variables before you run the command to start the utilities.

You can use the -FILE option when you start the following utilities: oninit, oncheck, onclean, ,onlog, onmode, onparams, onspaces, onstat.

Syntax

```
-FILE option

|--+---------+--+-----------+----------------------------------|
   '- -FILE=-'  '-file_name-'
```

Table 122. -FILE option

Element	Purpose	Key Considerations
-FILE=*file_name*	Specifies the full path or relative path to the `onconfig` file that contains the environment information.	The -FILE=*file_name* option must be the first argument in the command.

Usage

Before you run a command with the -FILE option, you must add directives to your `onconfig` file in the following format:

```
#$variable_name value
```

Any environment variables that are set in the `onconfig` file take precedence over the same environment variables that are set in the system or shell.

When you start a utility with the -FILE option, specify the full path or the relative path to the `onconfig` file. For example, both of the following examples start the database server with the environment information in the `onconfig.serv1` file:

Full path

```
oninit -FILE=/opt/HCL/inf/etc/onconfig.serv1
```

Relative path

```
oninit -FILE=etc/onconfig.serv1
```

If the ONEDB_HOME environment variable is not set in the user system, the shell, or in the `onconfig` file, the value of ONEDB_HOME is set to the PATH of the executable program, with the assumption that the executable program is in a subdirectory of ONEDB_HOME. For example, you can run the oninit -FILE=etc/onconfig.myserv command when the oninit utility is in the `/opt/HCL/onedb/bin` directory. If the ONEDB_HOME environment variable is not set in the shell or in the `onconfig.myserv` file, the value of ONEDB_HOME is set to `/opt/HCL/onedb`.

If you use a form of remote execution, such as `ssh`, use the -FILE option to specify the path to the `onconfig` file on the remote computer.

Example

Suppose that you specified values for the ONEDB_SERVER, DBDATE, and SERVER_LOCALE environment variables in the `onconfig` file for the js_3 instance:

```
              #onconfig.js_3
              #
              # *** Start environment settings for js_3
```

```
#
#$ONEDB_SERVER server3
#$DBDATE MDY4/
#$SERVER_LOCALE en_us.utf8
#
# *** End environment settings for js_3
```

The other important environment variables (ONEDB_HOME, ONEDB_ SQLHOSTS, ONCONFIG) for running the utility are specified in the user environment. The path to the oninit executable program is part of the user environment and the `onconfig` file is in the current directory.

You can run the `oninit -FILE=onconfig.js_3` command from the current directory to start the database server, and automatically set the values for the ONEDB_SERVER, DBDATE, and SERVER_LOCALE environment variables.

Return codes for the oninit utility

If a **oninit** command encounters an error, the database server returns an error message and a return code value.

The following table contains the return codes, message text, and user actions for the oninit utility.

Table 123. Return codes for the oninit utility

Return Code	Message Text	User Action
0	The database server was initialized successfully.	The database server started.
1	Server initialized has failed. Look at any error messages written to stderr or the online message log.	Take the appropriate action based on the error messages written to stderr or the online message log.
87	The database server has detected security violations or certain prerequisites are missing or incorrect.	(UNIX™ and Mac OS only) Check if user and group **informix** exists. Check if the server configuration file (`onconfig`) and `sqlhosts` file exists and has the correct permissions. Check if the environment variables **ONEDB_HOME**, **ONCONFIG**, and **SQLHOSTS** have a valid value and their length does not exceed 255 characters. Check if the environment variable **ONEDB_HOME** specifies an absolute path and does not have any spaces, tab, new lines, or other incorrect characters. Check if role separation-related subdirectories under the `$ONEDB_HOME` directory, such as `aaodir` and `dbssodir`, have the correct ownership. Run the onsecurity utility to diagnose and fix any issues.
170	The database server failed to initialize the dataskip structure.	Free some physical memory on the system and try to start the database server again.
172	The database server failed to initialize the listener threads.	Free some system resources, check the configuration parameter values for the number of listener threads to start

Table 123. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
		when the database server starts up, and try to start the database server again.
173	The database server failed to initialize data replication.	Free some physical memory in the system and try to start the database server again.
174	The database server failed to start fast recovery threads.	Free some physical memory in the system and try to start the database server again.
175	The database server failed to initialize the root dbspace.	Check the root dbspace related parameters in server configuration file (`onconfig`) to make sure that the path for the root dbspace is valid.
176	Shared disk secondary server initialization failed.	Check the entries in `sqlhosts` file (UNIX™) or SQLHOSTS registry key (Windows™) to make sure that you are using the value of the DBSERVERNAME configuration for the primary server correctly. Check if the value for the SDS_PAGING configuration parameter in the server configuration file (`onconfig`) is correct. Free some system resources and try to start the database server again.
177	The database server failed to start the main_loop thread.	Free some physical memory on the system and try to start the database server again.
178	The database server failed to initialize the memory required for page conversion.	Free some physical memory on the system and try to start the database server again.
179	The database server was unable to start CPU VPs.	Free some physical memory on the system and try to start the database server again.
180	The database server was unable to start the ADM VP.	Free some physical memory on the system and try to start the database server again.
181	The database server failed to initialize kernel AIO.	Free some physical memory on the system and try to start the database server again.
182	The database server was unable to start IO VPs.	Free some physical memory on the system and try to start the database server again.
183	The database server failed to initialize the memory required for asynchronous I/O operations.	Free some physical memory on the system and try to start the database server again.
184	The database server failed to initialize memory required for parallel database queries. (PDQ)	Free some physical memory on the system and try to start the database server again.

Table 123. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
185	The database server failed to initialize various SQL caches.	Free some physical memory on the system and try to start the database server again.
186	The database server failed to initialize the Global Language Support (GLS) component.	Free some physical memory on the system and try to start the database server again.
187	The database server failed to initialize the Associated Service Facility (ASF) components.	Check the entries in `sqlhosts` file.
188	The database server was unable to start the CRYPTO VP.	Free some physical memory on the system and try to start the database server again.
189	The database server was unable to initialize the alarm program.	Free some physical memory on the system and try to start the database server again.
190	The database server failed to initialize the auditing component.	Free some physical memory on the system and try to start the database server again.
192	The database server failed to restore the Window station and desktop.	(Windows™ only) Try to shut down the database server after freeing some system resources.
193	The database server failed to create daemon processes.	(UNIX™ and Mac OS only) Free some system resources and try to startup the database server once again.
194	The database server failed to redirect the file descriptors properly.	(UNIX™ and Mac OS only) Check the availability of the `/dev/null` device and try to start the database server again.
195	The database server failed to initialize the current directory for use.	Check the validity of the current working directory from where the database server is being initialized.
196	The database server failed to initialize the /dev/null device.	(AIX® only) Check the validity of the `/dev/null` device.
197	The database server failed to find the password information for the user trying to initialize the database server.	Verify that the user password is valid.
198	The database server failed to set the resource limits.	(UNIX™ and Mac OS only) Verify, and if required, increase the resource limits for processes on the host computer.
200	The database server did not have enough memory to allocate structures during initialization.	Free some physical memory on the system and try to start the database server again.
206	The database server could not allocate the first resident segment.	Check the values of the BUFFERPOOL and LOCKS configuration parameters in the server configuration file (`onconfig`) to

Table 123. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
		make sure that they can be accommodated with the available memory on the host computer.
207	The database server failed to initialize shared memory and disk space.	Free some physical memory in the system, check the validity of all the chunks in the database server, and try to start the database server again.
208	The database server failed to allocate structures from shared memory.	Free some system resources and try to start the database server again.
209	The database server encountered a fatal error during the creation of shared memory.	Free some physical memory in the system and try to start the database server again.
210	The database server requested memory for the resident segment that exceeded the maximum allowed.	Reduce the size of the resident segment by lowering the values of the BUFFERPOOL and LOCKS configuration parameters.
220	The database server failed to read the audit configuration file.	Check that the audit configuration file (`adtcfg`) exists and is valid.
221	The database server could not detect the default directory for DUMPDIR. Usually it is the $ONEDB_HOME/tmp directory.	Create the `$ONEDB_HOME/tmp` directory if it is not present.
222	The database server detected an error in the value of the DBSERVERALIASES configuration parameter in the server's configuration file.	Verify that the values for the DBSERVERALIASES configuration parameter are valid and they have corresponding entries in the `sqlhosts` file (UNIX™) or SQLHOSTS registry key (Windows™).
223	The database server detected an error with the value of the DBSERVERNAME configuration parameter in the server's configuration file.	Verify that the value of the DBSERVERNAME configuration parameter is valid and it has a corresponding entry in the `sqlhosts` file (UNIX™) or SQLHOSTS registry key (Windows™).
224	The database server detected an error with the value of the HA_ALIAS configuration parameter in the server's configuration file.	Correct the value of the HA_ALIAS configuration parameter in the server configuration file (`onconfig`).
225	The database server detected too many entries for the NETTYPE configuration parameter or the DBSERVERALIASES configuration parameter in the server's configuration file.	Reduce the number of instances of the NETTYPE or DBSERVERALIASES configuration parameters in server configuration file (`onconfig`) and try to start the database server again.
226	The database server could not find an entry for the DBSERVERNAME configuration parameter in the sqlhosts file or the contents of the sqlhosts file are not valid.	Check the entries in the `sqlhosts` file.

Table 123. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
227	Incorrect serial number.	Reinstall the database server.
228	The user does not have the necessary DBSA privileges to invoke the executable.	The user must have DBSA privileges or be a part of the HCL OneDB™-Admin group (Windows™).
229	The database server could not initialize the security sub-system.	(Windows™ only) The user does not the necessary user rights on the host or is not part of the HCL OneDB™-Admin group.
230	The database server, if started as a process on Windows™ platform , timed out while trying to build the required system databases during initialization. (Windows™ only)	Check the event log on the host to determine why the service could not be opened or could not be started. The database server might have timed out while trying to build the system databases. Free some system resources and try to start the database server again.
231	HCL OneDB™ service startup failed when the oninit -w command was run as a process on the command line.	(Windows™ only) Check the event log on the host to determine why the service start has failed.
233	The database server failed to initialize the Pluggable Authentication Module (PAM).	Check the configuration for the PAM library on the system.
235	The database server detected errors for certain configuration parameter values in the server's configuration file.	Inspect the server configuration file (`onconfig`) for any errors.
236	The database server detected an error while trying to restrict the allowable values for the HCL OneDB™ edition in use.	Check if the SDS_ENABLE configuration parameter is set to 1 in the server configuration file (`onconfig`). Check if the server name specified with the oninit -SDS command matches the value of the HA_ALIAS or DBSERVERNAME configuration parameter. Check if the shared disk used is part of an existing shared disk cluster.
237	The database server could not find the server configuration file.	Ensure that the server configuration file exists and is valid.
238	The database server detected an incorrect value for the ONEDB_SERVER environment variable or the value did not match the value of the DBSERVERNAME configuration parameter in the server's configuration file.	(Windows™ only) Check the value of the **ONEDB_SERVER** environment variable and the corresponding entry in the registry.
239	The database server detected an incorrect or non-existent value for the ONEDB_HOME environment variable.	(Windows™ only) Check the value of the **ONEDB_HOME** environment variable.

Table 123. Return codes for the oninit utility (continued)

Return Code	Message Text	User Action
240	Incorrect command-line options were issued to the database server.	Correct the command-line options issued to the database server at startup.
248	The database server failed to create the HCL OneDB™ loader domain file.	(AIX® only) Check if the `/var/adm/ifx_loader_domain` file is present.
249	The database server failed to dynamically load the PAM library.	The PAM library is not available for the database server. Install the PAM libraries.
250	The database server failed to dynamically load the ELF library.	The ELF library is not available to the database server. Install the libelf packages.
255	There was an internal error during server initialization. Look at any error messages written to stderr or to the online message log.	Take the appropriate action based on the error messages written to stderr or the online message log.

The onkstash Utility

Use the **onkstash** to create a password stash file for an existing PKCS#12 keystore.

A password stash file allows database clients or the database server itself access to their respective keystore without the inconvenience for the user to supply the password every time.

The **onkstash** utility accepts the file name of a PKCS#12 keystore (ending with extension ".p12") and the password for this keystore. It writes the password in an encrypted format to the password stashfile. The name of this stash file is same as the keystore filename, but with the extension ".stl".

If the password for a keystore gets changed, the new password must be stashed again using the **onkstash** utility. If a password stash file exist with the old keystore password, then it is overwritten with the new password in an encrypted format.

Syntax

```
onkstash <keystore file> <password>
```

where **<keystore file>** is the name of the PKCS#12 keystore file, and **<password>** is the current password for the keystore.

Usage

The **onkstash** utility determines the file name for the password stash file from the name of the keystore file. It checks if the given password is correct and then writes it in an encrypted format to the stash file.

If the password stash file gets created by **onkstash**, the file access permissions are set to 600. If the password stash file already exist, the permissions are not changed. It is recommended to check the permissions for the keystore file as well as for the password stash file, and correct them if deemed necessary.

The onkstore Utility

Use the **onkstore** utility to create and manage password stash files for use with storage space encryption and the integrated backup encryption features.

The onkstore utility will create a password stash file in the $ONEDB_HOME/etc directory by default, but this file may be created and used from any location accessible by the database server as long as that directory has secure permissions.

With its informix/informix ownership and 600 permissions, the password stash file can be read only by users root or informix in UNIX/Linux and the creator of the keystore in Windows. In addition, the file is itself encrypted using a password. The admin must specify this *keystore password* when creating the password stash file. By default that password will be stored (as an obfuscated value) in a stash file along side the password stash file. Do not remove the stash file or allow it to be separated from the password stash file. If you do not want the password to be stashed, use the option "-nostash" when creating the keystore. In that case the password may be supplied interactively to oninit and utilities such as *oncheck*, *onlog*, or *onbar*.

The onkstore utility can create different types of password stash files. A password stash file can contain either:

1. A *Master Encryption Key* (MEK) that is used as a "seed? by the server to encrypt storage spaces when using it with the Storage Space Encryption feature.
2. A set of credentials to access a Remote Key Server that stores the Master Encryption Key for the Storage Space Encryption (DISK_ENCRYPTION configuration parameter on page 79) or a set of credentials to access a Remote Key Server that stores the Remote Master Encryption Key used by the Integrated Backup Encryption feature (BAR_ENCRYPTION configuration parameter).

The onkstore utility has the following usage:

Table 124. onkstore usage

-file <fn>	name of keystore to create/list/convert.
-type	type of keystore to create: local, AWS-EAR, AWS-BAR, KMIP, AZURE-EAR, AZURE-BAR
-create	create a new keystore. By default stash the password in a stash file. Use option "-nostash" if this is not desired.
-pw <fn>	file with cleartext keystore password. If not provided and the password is not stashed already, it is prompted for interactively.
-list	list the contents of the file.
-cipher	cipher the server will use: aes128, aes192, aes256

Table 124. onkstore usage (continued)

-credential <fn>	file that contains credentials in json format.
-pw [<fn>]	Current password for the keystore, supplied either interactively or in a file.
-verify	verify the keystore.
-convert	convert keystore from one type to another.
-changepw [<fn>]	change the password for the keystore.
-nostash	upon creation of a keystore do not stash the password.
-help	print this message.

Note: -pw is not needed if your password is stashed.

Use the onkstore utility to perform the following tasks:

Create a Keystore with onkstore

A password stash file is required by any instance that has the storage space encryption feature enabled. This password stash file has a ".p12? extension. It may also have an associated stash file whose extension is ".sth?.

When referring to a password stash file with onkstore or in the value of the DISK_ENCRYPTION configuration parameter, always omit the ".p12? extension.

A password stash file that contains your instance's encryption key is called a local password stash file. The simplest way to create a local password stash file is as follows:

```
onkstore -create -file my_keystore -type local -cipher aes128
```

The result of that command is a file located in the `$ONEDB_HOME/etc` directory called `my_keystore.p12`, which contains a 128-bit (16 byte) encryption key. That p12 file is encrypted using a password, which must be provided interactively when prompted for. By default, the password is stored in a stash file. The path to the stash file is `$ONEDB_HOME/etc/my_keystore.sth`.

To explicitly set a password for the new password stash file, create the file using this command instead:

```
echo "sample_password" > pw_file onkstore -file my_keystore -type local -cipher aes128 -pw pw_file rm pw_file
```

The password must be at least 8 characters long. In this case "sample_passwd? would also be stashed encrypted in `$ONEDB_HOME/etc/my_keystore.sth.`

As the encryption password is known, the admin has the option of removing the stash file and supplying the password to **oninit** manually each time the server is booted:

```
oninit -pw
Please enter current encryption password: sample_password
```

Instead of supplying the password interactively, it may be passed to oninit using a file:

```
touch /tmp/mypassword
chmod 660 /tmp/mypassword
echo “sample_password? > /tmp/mypassword
oninit -pw /tmp/mypassword
rm /tmp/mypassword
```

The password stash file will be located in `$ONEDB_HOME/etc` by default, but you can also move or create it elsewhere by specifying a full path (minus the .p12 extension):

```
onkstore -create -file /work/KEYSTORES/my_keystore -type local -cipher aes128
```

If your password stash file is not located in `$ONEDB_HOME/etc` you must use the full path in your DISK_ENCRYPTION setting:

```
DISK_ENCRYPTION keystore=/work/KEYSTORES/my_keystore
```

Like $ONEDB_HOME/etc, the directory containing your password stash file must have ownerships of informix/informix.

When creating a password stash file with onkstore you must specify which of the three supported ciphers you wish to use: aes128, aes192, and aes256. By default the server assumes you are using aes128, but if not, the admin must specify the cipher in the DISK_ENCRYPTION setting:

```
DISK_ENCRYPTION keystore=my_keystore,cipher=aes256
```

The DISK_ENCRYPTION setting consists of comma-separated attributes and may contain no quotes or spaces.

A password stash file that contains AWS (Amazon Web Services) credentials instead of an encryption key is called a remote password stash file. Run the following command to create a remote password stash file interactively:

```
onkstore -create -file my_aws_keystore -type AWS_EAR -cipher aes192
```

onkstore will then prompt you for AWS credentials and other information that will identify the key you want to either create or use. For example:

```
 $ onkstore -create -file my_aws_keystore -type AWS_EAR -cipher aes192
Creating AWS EAR Keystore
AWS Key Id
>AKCAIPP520LF4AJBOTXA
AWS Key Secret
>TCEmlasjdflkjbasNHFAI6BHOwj4XHe50ic7LCt9
AWS Region
>us-east-1
AWS CMK Id
>16fd15d9-db8b-4cb7-9d99-d3070df97b58
SSM Key Location
>/informix/keys/aes192/key1
```

This is not your actual encryption key. They are merely pieces of information that when put together allow the server to access a particular encryption key stored in AWS. If the terms “CMK Id? and “AWS Region? are not familiar to you, it is because you do not yet have an AWS account set up. Familiarity with an AWS account you are able to manage is a prerequisite for creating a remote password stash file using onkstore.

Rather than providing these details to onkstore interactively you have the option of feeding a json file to the utility instead:

```
onkstore -create -file my_ks -cipher aes192 -credential /tmp/my_creds.json
```

In this case the `/tmp/my_creds.json` file would contain something like this:

```
{
"Credentials" :
{
"Type" : "aws-ear",
"AWS Key Id" : "AKCAIPP520LF4AJBOTXA",
"AWS Key Secret" : "TCEmlasjdflkjbasNHFAI6BHOwj4XHe50ic7LCt9",
"AWS Region" : "us-east-1",
"AWS CMK Id" : "16fd15d9-db8b-4cb7-9d99-d3070df97b58",
"SSM Key Location" : "/informix/keys/aes192/key1"
}
}
```

If this command is run and the master encryption key does not exist in AWS at the specified location (/informix/keys/aes192/key1), onkstore will attempt to generate one and store it there. If the credentials point to an existing key, onkstore will create the password stash file and leave the key as-is.

The -pw argument works the same way with remote password stash file creation as it does with local keystore creation.

Do not use the AWS-BAR type when creating a keystore for use with the storage space encryption feature. This type of keystore is used with the Integrated Backup Encryption feature.

Creating an AWS type keystore

If your remote key server is Amazon Web Services Key Management Service (AWS-KMS), you can create two types of keystore : “AWS-EAR? to be used by the Storage Space Encryption feature, or “AWS-BAR? to be used by the Integrated Backup Encryption feature.

The only difference between this credentials is that the AWS-EAR requires, also, access to the AWS Secrets Manager (AWS-SSM) where the IDS Master Encryption Key is stored.

When asked to create a AWS keystore the following information must be readily available by the operator:

- AWS Key Id, this is not an encryption key, this is the AWS access key (the equivalent of a username) to get access to the AWS infrastructure. You can generate a “AWS Key Id?/?AWS Key Secret? pair (**AWS Services -> IAM -> Users -> “User Name? -> Security Credentials -> Access Keys**).
- AWS Key Secret, this is not an encryption key, this is the AWS secret key (the equivalent of a password) to get access to the AWS infrastructure. You can generate a “AWS Key Id?/?AWS Key Secret? pair (**AWS Services -> IAM -> Users -> “User Name? -> Security Credentials -> Access Keys**).
- AWS Region, This is the AWS region, all keys and CMKs are region bound, you must provide the region where you have your keys (ie us-east-1).

- AWS CMK Id, This is the id (or name) of the AWS Customer Master Key (Remote Master Encryption Key). This key never leaves the AWS infrastructure and it is used by onkstore to generate the master encryption key used by IDS with the Storage Spaces Encryption feature or the Backup Encryption Keys used by the Integrated Backup Encryption feature.
- SSM Key Location, This is needed only for AWS-EAR types of keystores. This is hierarchical path where the IDS Master Encryption Key will be stored after being generated by onkstore. The MEK is encrypted using the CMK before being stored here.

To use a JSON file as input for onkstore, create a file with the following structure:

```
{
  "Credentials" :
    {
    "Type" : "..." ,
    "AWS Key Id" : "...",
    "AWS Key Secret" : "...",
    "AWS Region" : "...",
    "AWS CMK Id" : "...",
    "SSM Key Location" : "..."
    }
  }
where the value for "Type" is either "aws-ear" or "aws-bar".
```

Creating an Azure type keystore

If your remote key server is Microsoft Azure Key Vault you can create two types of keystore : “AZURE-EAR? to be used by the Storage Space Encryption feature, or “AZURE-BAR? to be used by the Integrated Backup Encryption feature.

The only difference between this credentials is that the AZURE-EAR requires, also, access to the Azure Secrets inside the Key Vault, where the IDS Master Encryption Key is stored.

When asked to create a AZURE keystore, the following information must be readily available by the operator:

- Azure Vault Url, this is the URL that access your key vault. This value is generated by the Azure system when a new keyvault is created (**Home -> All resources -> “Key Vault Name? -> Overview -> DNS Name**).
- Azure Client Id, the usage of this feature requires access to the Active directory infrastructure in Azure,for that you need to create a Web Application under your username and provide the “Application Id? (**Active Directory -> Users -> All User -> “User Id? -> Applications -> “Application Name? -> Application Id**).
- Azure Client Secret, When the Web Application is created, you will be provided with both the Application Id and the Application Secret. The application secret cannot be recovered after the application was created.
- Azure Directory Id, Your KeyVault is created under an Active Directory, you need to provide the Directory Id (**Home -> All resources -> “Key Vault Name? -> Overview -> Directory ID**).
- Azure Key Name, this is the name or full id of the Azure Key (Remote Master Encryption Key). This Key never leaves the Azure infrastructure and it is used by onkstore to encrypt the locally generated Master Encryption Key used by the IDS Storage Space Encryption feature. It is also used by the On-Bar utilitiy to encrypt the Backup Encryption Keys used by the Integrated Backup Encryption Feature. In Azure, you can provide a simple name for this key (ie “MY_IDS_MEK?) in which case we will use the LATEST key available (Each time the key

is rotated a newer Id is available), or, you can specify the Id of the key you want to use (ie " MY_IDS_MEK/wsdd6405fb584cf9a3c63f6926d2e92e?) in which case we will keep using the same key even if it is rotated.
- Azure Encrypt Algorithm, when you create the RMEK in Azure Key Vault, it allows you to select among several types of keys and depending on the type of key, you can select different algorithms to encrypt data with it. Select here a valid algorithm name for the type of RMEK you created.
- Azure Secret Name, The name of the secret where we will store the IDS MEK. This is used only if you create a AZURE-EAR type keystore. If you provide a simple name (ie " INFORMIX-256BIT?) a new MEK will be generated and stored, the ID of the newly stored key will recorded. If you provide a full ID for the secret (" INFORMIX-256BIT/284ded569a8b40be8e4de2254ddeedd7?), then we will try to retrieve the secret, if not present we will return an error.

To use a JSON file as input for onkstore, create a file with the following structure:

```
{
  "Credentials" :
    {
    "Type" : "...",
    "Azure Vault Url" : "...",
    "Azure Client Id" : "...",
    "Azure Client Secret" : "...",
    "Azure Directory Id" : "...",
    "Azure Key Name" : "...",
    "Azure Encrypt Algorithm" : "...",
    "Azure Secret Name" : "..."
    }
}

where the value for "Type" is either "azure-ear" or "azure-bar".
```

Creating a KMIP type keystore

If your remote key server is located in a server/cluster supporting the KMIP standard you can create a single type of keystore (KMIP). At this moment, the same keystore type can be used by both the Storage Space Encryption and Integrated Backup Encryption features.

For Integrated Backup Encryption, this type of keystore works similarly with Azure and AWS: We provide the Key name of a RMEK that is used to encrypt the Backup Encryption Keys.

For Storage Space Encryption, the Key Name provided is the IDS MEK.

When asked to create a KMIP type keystore the following information must be readily available by the operator:

- KMIP Server, the IP address or hostname where the KMIP server is listening for request. If the port where the server listens is different from the default (5696), the port must be specified (ie "myserver.hcl.com:2356).
- KMIP Username, username to access the KMIP server. This is optional since in most cases, the access to the server is done by using SSL certificates.
- KMIP Password, password for the given username. This is also optional.
- KMIP Client Certificate File, a file containing the certificate for the client, The file must also contain the Private Key matching the certificate. The private key is expected to be a PKCS#8 key. The certificate is expected to have Authentication extensions.

- KMIP CA Certificate File, a file containing the root CA used to sign both the KMIP Client Certificate File and the KMIP Server Certificate File.
- KMIP Key Name, The name of the KMIP Key used as MEK by the Storage Spaces Encryuption feature or as RMEK by the Integrated backup Encryption Feature. It is optional. If not present, onkstore will generate a new key and report its Id to the operator.

To use a JSON file as input for onkstore, create a file with the following structure:

```
{
              "Credentials" :
                {
                "Type" : "..." ,
                "KMIP Server" : "...",
                "KMIP Username" : "...",
                "KMIP Password" : "...",
                "KMIP Client Certificate File" : "...",
                "KMIP CA Certificate File" : "...",
                "KMIP Key Name" : "..."
                }
            }

          where the value for "Type" is "kmip".
```

Verifying a Keystore File

After creating a network keystore file and before deploying it, it is important to verify that its credentials work correctly. To do this, run the following command:

onkstore -file my_keystore -verify

onkstore will use the credentials contained in your keystore file to communicate with AWS and report success or failure:

```
$ onkstore -file my_keystore -verify
Keystore Verify Successful.
Key exists in AWS SSM /informix/keys/aes192/key1 for cipher aes192.
```

Changing the Password for a Keystore File

At any time using onkstore the admin may change the password of a keystore:

onkstore -file my_keystore -pw /tmp/old_password -changepw /tmp/new_password

If your current password is stashed (contained in the keystore’s associated .sth file), then you do not need to pass it to onkstore via the -pw argument, but if you have removed the stash file you must provide the current password to onkstore before it can be changed to a new one.

You can also perform this same function using the “master_key reset"? sysadmin command:

```
dbaccess sysadmin -<<END
execute function task(“master_key reset"?new_sample_pw?);
END
```

This method has the advantage of not requiring the current password in order to change to a new one. The current password was provided to the server at boot time.

Converting a Keystore File

The convert feature is currently used only for EAR types of keystores. It supports to download the Master Encryption Key contained in the Remote Key Server (ie a KMIP server) to the local keystore. The old keystore containing the credentials to the RKS will be renamed and will be replaced with a new one of type “local?.

Since the Integrated Backup Encryption feature does not store a Master Encryption Key at the RKS and does not support keystore of type “local?, this option is not needed/supported for credentials of type AWS-BAR and AZURE-BAR.

```
$ onkstore -file my_keystore -convert
Which type of keystore would you like to create:
1 - Local Keystore
2 - AWS EAR Keystore
3 - AWS BAR Keystore
4 - KMIP EAR Keystore
5 - AZURE EAR Keystore
6 - AZURE BAR Keystore

Conversion complete for /vobs/tristarp/sqldist/etc/my_keystore.p12
```

Currently, only option 1 (converting to a local keystore file) is supported. The original keystore file is copied to a backup file (my_keystore.p12.bak#) before being overwritten during the conversion.

Note: By downloading your MEK to a local machine, you are increasing the chances of exposing that key, which is the reason to use a RKS in the first place.

List the contents of a Keystore File

This command will not display your encryption key or your AWS credentials. It displays the kinds of objects stored in the file. For example:

```
$ onkstore -file /work3/keystores/test_keystore -list
List the contents of keystore /work3/keystores/test_keystore.p12
KeystoreType
AWS Key Id
AWS Key Secret
AWS Region
AWS CMK Id
SSM Key Location
```

An admin may list the basic contents of a file as a sort of sanity check.

The onlog utility

The **onlog** utility displays the contents of a logical-log file, either on disk or on backup.

onlog: Display Logical-Log Contents

The **onlog** output is useful in debugging situations when you want to track a specific transaction or see what changes have been made to a specific tblspace. (For information about interpreting the logical-log file contents, see Interpreting Logical-Log Records.)

Any user can run all of the **onlog** options except the **-l** option. Only user **informix** on UNIX™ or a member of the **Informix-Admin** group on Windows™ can run the **-l** option.

If the database server is in offline mode when you execute **onlog**, only the files on disk are read. If the database server is in quiescent or online mode, **onlog** also reads the logical-log records stored in the logical-log buffers in shared memory (after all records on disk have been read).

When the database server reads a logical-log file with status `U` from disk while in online mode, the database server denies all access to the logical-log files, effectively stopping database activity for all sessions. (For more information, see onstat -l command: Print physical and logical log information on page 681.) For this reason, it is recommended that you wait until the files have been backed up and then read the contents of the logical-log files from backup.

The **onlog** utility does not have a functionally equivalent SQL administration API *command* string.

onlog Syntax

```
>>-onlog--+------------------------+--+----------------------+-->
          |                     (1) |  '- -pw-+------------+--'
          '-|  -FILE option   |-----'          '--filename--'

                                (2)                                 3)
>--| Log-Record Read Filters  |-----| Log-Record Display Filters |------>

>--+-----+--+-----------+--------------------------------------><
   '- -q-'  +- -V-------+
            '- -version-'
```

Note: The -pw option is required only when the Storage space encryption feature is enabled, stash file is in use, and the server is off-line. Supply an optional path to a file containing the keystore password, otherwise onlog will prompt for a password before displaying the requested log(s).

Element	Purpose	Key Considerations
-q	Suppresses the initial header and the one-line header that appears every 18 records by default	None.
-V	Displays the software version number and the serial number	See Obtaining utility version information on page 312.

Element	Purpose	Key Considerations
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See Obtaining utility version information on page 312.

You direct **onlog** to read the following portions of the logical log as it searches for records to display:

- Records stored on disk
- Records stored on backup media
- Records from the specified logical-log file

By default, **onlog** displays the logical-log record header, which describes the transaction number and the record type. The record type identifies the type of operation performed.

In addition to the header, you can use the read filters to direct **onlog** to display the following information:

- Logical-log record header and data (including copies of simple large objects stored in a dbspace or tblspace)
- Copies of blobpages from blobspaces

  They are copied from the logical-log backup only. They are not available from disk.

You can display every logical-log record header, or you can specify output based on the following criteria:

- Records associated with a specific table
- Records initiated by a specific user
- Records associated with a specific transaction

If **onlog** detects an error in the log file, such as an unrecognizable log type, it displays the entire log page in hexadecimal format and terminates.

Log-Record Read Filters

The **onlog** utility uses the pathnames that are stored in the root dbspace reserved pages to locate the logical-log files. If you use ON-Bar to back up the logical logs, **onlog** asks the storage manager to retrieve the appropriate logical-log records from the backup media.

```
Syntax

|--+----------------------+------------------------------------->
   '- -d--device--+-----+-'
                  '- -b-'

   .--------------------------------------------.
   V                                            |
>----+---------------------------------------+-+----------------|
     '- -n--starting_uniqid - ending_uniquid-'
```

Element	Purpose	Key Considerations
-b	Displays logical-log records associated with blobspace blobpages	The database server stores these records on the logical-log backup media as part of blobspace logging.
-d *device*	Names the pathname of the storage device where the desired logical-log backup is mounted	If the **-d** option is not used, **onlog** reads the logical-log files stored on disk, starting with the logical-log file with the lowest *logid*. If you use ON-Bar to back up logical logs, use the **onbar -P** command to view the contents of a logical-log file. For pathname syntax, see your operating-system documentation.
-n *starting_uniqid-ending_uniqid*	Directs **onlog** to read all the logical-log records contained in the log file that you specified from *starting uniqid* to the *ending uniqid*.	The *starting_uniqid* and the *ending_uniqid* are the unique ID numbers of the logical log. To determine the *uniqid* of a particular logical-log file, use the **onstat -l** command. If you do not use the **-n** option, **onlog** reads all the logical-log files that are available (either on disk or on tape). For information about the **onstat** utility, see Monitor the database server status on page 478.

Log-Record Display Filters

```
Syntax

   .-------------------------------------------.
   V                                           |
|----+---------------------------------------+-+-----------------|
     |     (1)                               |
     +- -l-----------------------------------+
     |                   (1)                 |
     +- -t--tblspace_num---------------------+
     |             (1)                       |
     +- -u--username-------------------------+
     |                   (1)                 |
     +- -x--transaction_id-------------------+
     |                               (1)     |
     '- -c--compression_dictionary_file-----'
```

Element	Purpose	Key Considerations
-l	Displays the long listing of the logical-log record.	The long listing of a log record includes a complex hexadecimal and ASCII dump of the entire log record. The listing is not intended for casual use.
-**t***tblspace_num*	Displays records associated with the tblspace that you specify.	Unsigned integer. Number, greater than 0, must be in the **partnum** column of the **systables** system catalog table. Specify this value as either an integer or hexadecimal value. (If you do not use a 0x prefix, the value is interpreted as an integer.) To determine the tblspace number of a particular tblspace, query the **systables** system catalog table as described in Tblspace Numbers on page 287.
-**u** *username*	Displays records for a specific user.	User name must be an existing login name. User name must conform to operating-system-specific rules for login name.
-**x** *transaction_id*	Displays only records associated with the transaction that you specify.	Value must be an unsigned integer between 0 and TRANSACTIONS - 1, inclusive. You should need to use the **-x** option only in the unlikely case that an error is generated during a rollforward. When this situation occurs, the database server sends a message to the message log that includes the transaction ID of the offending transaction. You can use this transaction ID with the **-x** option of **onlog** to investigate the cause of the error.
-c *compression_ dictionary_file*	Uses the compression dictionary to expand compressed data and display uncompressed data.	If the **onlog** command contains the **-l** option and the **-c** option and there are compressed images in the log records, the **onlog** utility uses the compression dictionary to expand all expandable images in the log records. A compressed image is expandable only if there is a valid compression dictionary for that log record in the compression dictionary file. If **-c** is not specified or the compression dictionary file does not contain a valid compression dictionary for the compressed image, the **onlog** utility will display the row image in its compressed format.

If you do not have a compression dictionary file, you can use an UNLOAD statement to unload the compression dictionary, which is contained in the **syscompdicts_full** table in the **sysmaster** database, to a compression dictionary file, as follows:

```
UNLOAD TO 'compression_dictionary_file'
   SELECT * FROM sysmaster:syscompdicts_full;
```

If you do not specify any options, **onlog** displays a short listing of all the records in the log. You can combine options with any other options to produce more selective filters. For example, if you use both the **-u** and **-x** options, **onlog** displays only the activities that the specified user initiated during the specified transaction. If you use both the **-u** and **-t** options, **onlog** displays only the activities initiated by the specified user and associated with the specified tblspace.

The onmode utility

Use the **onmode** utility to change the database server operating mode and perform various other operations on shared memory, sessions, transactions, parameters, and segments.

These topics show how to use the **onmode** options. If you do not use any options, the database server returns a usage statement.

On UNIX™, you must be user **root** or user **informix** to run the **onmode** utility.

On Windows™, you must be a member of the **Informix-Admin** group or the Administrators group to run the **onmode** utility.

For information on the **onmode -b** command, which is only used if you upgraded to a new version of OneDB and need to revert your databases to the previous version of the server, see Syntax of the onmode -b command on page in the *HCL OneDB™ Migration Guide*.

All **onmode** command options have equivalent SQL administration API *command* strings, except **onmode -b**, **onmode -BC**, and **onmode -R**.

onmode command syntax

Use **onmode** utility commands to perform various database server operations.

The following syntax diagram shows all of the options that you can use with the onmode command. The syntax diagram does not show all of the elements that you use with each command option. For the complete syntax of each command, see the topic on that command.

Syntax

```
>>-onmode--+-------------------------+------------------------->
           |                     (1) |
           '-|  -FILE option   |-----'

>--+- -a---------------+--+-----------+--+-----+-------------><
   |     (2)           |  +- -V-------+  '- -y-'
   +- -b---------------+  '- -version-'
   +- -BC--------------+
   +- -C---------------+
   +- -c---------------+
   +- -cache surrogates-+
   +- -e---------------+
   +- -d---------------+
   +-+- -D-+-----------+
   | +- -M-+           |
   | +- -Q-+           |
   | '- -S-'           |
```

```
   +- -F----------------+
   +- -h-+---------+----+
   |      '- force -'    |
   +- -I----------------+
   +-+- -j-+------------+
   | +- -k-+             |
   | +- -m-+             |
   | +- -s-+             |
   | '- -u-'             |
   +- -l----------------+
   +-+- -n-+------------+
   | '- -r-'             |
   +- -O----------------+
   +- -P----------------+
   +- -p----------------+
   +- -R----------------+
   +- -W----------------+
   +-+- -wf-+-----------+
   | '- -wm-'            |
   +-+- -we-+-----------+
   | '- -wi-'            |
   +- -Y----------------+
   +- -Z----------------+
   '- -z----------------'
```

Element	Purpose	Key Considerations
-y	Causes the database server to automatically respond yes to all prompts	None.
-V	Displays the software version number and the serial number	See Obtaining utility version information on page 312.
-version	Displays the build version, host, OS, number and date, as well as the GLS version	See Obtaining utility version information on page 312.

onmode -a: Add a shared-memory segment

```
>>-onmode-- -a--seg_size---------------------------------------><
```

Element	Purpose	Key considerations
-a *seg_size*	Allows you to add a new virtual shared-memory segment. Size is specified in kilobytes	**Restriction:** The value of ***seg_size*** must be a positive integer. It must not exceed the operating system limit on the size of shared-memory segments.

Ordinarily, you do not need to add segments to the virtual portion of shared memory because the database server automatically adds segments as they are needed. However, as segments are added, the database server might reach the operating-system limit for the maximum number of segments before it acquires the memory that it needs. This situation typically occurs when the SHMADD configuration parameter is set so small that the database server exhausts the number of available segments before it acquires the memory that it needs for some operation.

You can use this command to add a segment that is larger than the size specified by the SHMADD configuration parameter. By using this command to add a segment, you can adhere to the operating system limit for segments while meeting the need that the database server has for additional memory.

This command has an equivalent SQL administration API function.

onmode -BC: Allow large chunk mode

```
>>-onmode--+- -BC 1-+-----------------------------------------><
           '- -BC 2-'
```

Element	Purpose	Key Considerations
-BC 1	Enables support of large chunks, large offsets that are greater than 2 GB, and allows up to 32,768 chunks per instance.	This option allows large chunks to be created. Reversion without dropping the dbspace is possible if no chunks are larger than 2 GB. Dbspaces and blobspaces without chunks greater than 2 GB remain in the old format. After a chunk larger than 2 GB is added to a dbspace or blobspace then all chunks added or altered in that dbspace or blobspace are in the new format. See your *HCL OneDB™ Administrator's Guide*.
-BC 2	Allows large-chunk-only mode for all dbspaces.	Reversion is not possible. Enables the 9.4 large chunk feature for all dbspaces and blobspaces, Any chunk or offset added or modified has the new format. Existing chunks that you do not alter remain in the old format. See your *HCL OneDB™ Administrator's Guide*.

The **onmode -BC** (backward-compatible) commands are useful if you have converted from HCL OneDB™ 9.40 (small chunk mode) to HCL OneDB™ 10.0 or later. When HCL OneDB™ 10.0 or later is first initialized (with the **oninit -iyv** command), by default it comes online with large chunk mode already fully enabled. Reversion is not possible. In the case of a newly initialized instance of HCL OneDB™ 10.0 or later, the **onmode -BC** commands will return an error.

Note: After executing the **onmode -BC** command, perform a complete system level-0 backup.

onmode -c: Force a checkpoint

```
>>-onmode-- -c--+--------------------+------------------------><
                |        .- 15 ----. |
                +-block--+---------+-+
                |        '-timeout-' |
```

```
                '-unblock------------'
```

Element	Purpose	Key considerations
-c	Forces a checkpoint that flushes the buffers to disk.	You can use the **-c** option to force a sync checkpoint if the most recent checkpoint record in the logical log was preventing the logical-log file from being freed (status U-B-L).
block	Blocks the database server from any transactions.	While the database server is blocked, users can access it in read-only mode. Use this option to perform an external backup on HCL OneDB™. For more information, see the *HCL OneDB™ Backup and Restore Guide*.
timeout	Specifies the number of seconds to wait for checkpoints to clear before returning to the command prompt.	The *timeout* option applies only if the DELAY_APPLY configuration parameter is configured (see DELAY_APPLY Configuration Parameter on page 75. If the DELAY_APPLY configuration parameter is enabled, the checkpoint requested by the primary server might not arrive at the secondary server for an extended period of time. It is also possible that no other checkpoints are staged in the staging directory. The default timeout value is 15 seconds and the maximum timeout allowed is 10 minutes (600 seconds). See *HCL OneDB™ Backup and Restore Guide*.
unblock	Unblocks the database server.	When the database server is unblocked, data transactions and normal database server operations can resume. Use this option after you complete an external backup on HCL OneDB™. For more information, see the *HCL OneDB™ Backup and Restore Guide*.

This command has an equivalent SQL administration API function.

onmode -C: Control the B-tree scanner

Use the onmode -C command to control the B-tree scanner and specify information about B-tree scanner threads.

```
>>-onmode-- -C--+-(yielding syntax)--+------------------------><
                +-start--------------+
                +-count--------------+
                +-stop--count--------+
                +-kill    -count-----+
                +-threshold--size----+
                +-duration--num------+
                +-rangesize--size----+
                +-alice--mode--------+
                '-compression--value-'
```

Element	Purpose	Key considerations
-C	Controls the B-tree scanner for cleaning indexes of deleted items	There is no limit to the number of threads that can run at one time. However, there is a limit of 128 threads that can be started at one time. If, for example, you wanted 150 threads to run, you could execute two commands: onmode -C 100 and onmode -C 50.
start *count*	Starts additional B-tree scanner threads.	If *count* is not specified, a *count* of 1 is assumed. There is no limit on the number of scanner threads that can be specified.
stop *count* **kill** *count*	Stops B-tree scanner threads.	If *count* is not specified, a *count* of 1 is assumed. Stopping all index scanners prevents all index cleaning. Either of these commands stop the B-tree scanner.
threshold size*count*	Sets the minimum number of deleted items an index must encounter before an index is placed on the hot list.	Once all indexes above the threshold have been cleaned and there is no other work for the B-tree scanner to do, the indexes below the threshold are added to the hot list.
duration *num*	The number of seconds that the hot list is valid.	After this number of seconds expires, the hot list will be rebuilt by the next available B-tree scanner thread, even if unprocessed items are on the list. Scanners currently processing requests are not interrupted.
rangesize *size*	Determines the size of an index before index range cleaning is enabled.	A size of `-1` can be used to disable range scanning.
alice *num*	Sets the system's **alice** mode.	Valid *num* values range from `0` (`OFF`) to `12`.
compression *value*	For a database server instance, modifies the level at which two partially used index pages are merged. The pages are merged if the data on those pages totals a set level.	Valid values for the level are `low`, `med` (medium), `high`, and `default`. The system default value is `med`.

The B-tree scanner has statistical information which tracks index efficiency and how much extra work the index currently places on the server. Based on the amount of extra work the index has accomplished because of committed deleted index items, the B-tree scanner develops an ordered list of indexes that have caused the server to do extra work. This list is called the hot list. The index causing the highest amount of extra work is cleaned first and the rest of the indexes are cleaned in descending order. The DBA can allocate cleaning threads dynamically, thus allowing for configurable workloads.

This command has an equivalent SQL administration API function.

onmode -cache surrogates: Cache the allowed.surrogates file

```
>>-onmode-- -cache surrogates--------------------------------><
```

Element	Purpose	Key considerations
-cache surrogates	Reads the `/etc/onedb/allowed.surrogates` file and stores the user IDs and group IDs values in shared memory cache. The user names and group names specified in `allowed.surrogates` file have to be valid operating system users and groups. The names are converted to corresponding UIDs and GIDs.	You can use onmode -cache surrogates during a session to load the `allowed.surrogates` file. The `allowed.surrogates` file is used specify users and groups who can act as surrogates for mapped users. The `allowed.surrogates` file will be automatically checked before a new connection is made to the database server or when users are created or altered. If the cache-refresh fails, the existing surrogate cache is cleared, effectively disabling mapped users. Existing connections on the server will be unaffected by changes in shared-memory cache. Changes in shared memory cache affect new sessions.

onmode -d: Set data-replication types

```
>>-onmode-- -d--+-standard---------------+--------------------><
                '-+-primary---+--ha_alias-'
                  '-secondary-'
```

Element	Purpose	Key Considerations
-d	Used to set a server's data-replication type.	You can use the -d standard option when the database server is in quiescent, online, or read-only mode.
ha_alias	Identifies the high-availability alias of the primary or secondary database server.	The high-availability alias is the server's HA_ALIAS configuration parameter value. The *ha_alias* argument of the other database server in the data-replication pair and the database server's type (standard, primary, or secondary) is preserved after reinitialization of shared memory.

Using the -d standard option

The -d standard option drops the connection between database servers in a data replication pair (if one exists) and sets the database server type of the current database server to standard. This option does not change the mode or type of the other database server in the pair.

Use the onmode -d standard command only to disconnect a primary server from an HDR secondary server. Running the command converts the HDR secondary server to a standalone server. You should not run the onmode -d standard command to disconnect a primary server from an RS secondary server. To disconnect a primary server from an RS secondary server run the following commands:

On the RS secondary server:

```
onmode -d standard
```

On the primary server:

```
onmode -d delete RSS rss_ha_alias
```

Using the -d primary option

The -d primary option sets the database server type to primary and attempts to connect with the database server that *dbservername* specifies. If the connection is successful, data replication is turned on. The primary database server goes into online mode, and the secondary database server goes into read-only mode. If the connection is not successful, the database server is in online mode, but data replication is not turned on.

Using the -d secondary option

The -d secondary option sets the database server type to secondary and attempts to connect with the database server that *ha_alias* specifies. If the connection is successful, data replication is turned on. If the primary database server goes online, and the secondary database server goes into read-only mode. If the connection is not successful, the database server is in read-only mode, but data replication is not turned on.

This command has an equivalent SQL administration API function.

onmode -d: Set High Availability server characteristics

```

                                       (1)
>>-onmode -d--+-make primary--ha_alias------+-------+-+-------->< 
              |                             '-force-' |
              +-| RS Secondary server commands |------+
              '-| SD Secondary server commands |------'

RS Secondary server commands

                             (2)
|--+-+-add RSS--rss_ha_alias-----+--+----------+-+--------------|
   | |                       (1) |  '-password-' |
   | '-RSS--primary_ha_alias-----'               |
   |                                    (2)      |
   '-+-change RSS--rss_ha_alias--password-----+--'
     |                          (2)          |
     '-delete RSS--rss_ha_alias---------------'

SD Secondary server commands

                               (3)
```

```
|--+-set SDS primary--ha_alias------+-------+-+-----------------|
   |                                '-force-' |
   |                                     (2)  |
   '-clear SDS primary--primary_ha_alias------'
```

Element	Purpose	Key Considerations
-d	Used to create, modify, or delete secondary servers in high-availability configurations	
add RSS	Adds an RS secondary server	This command should be run on the primary database server.
rss_ha_alias	Identifies the RS secondary database server's high-availability alias.	The value can be an HA_ALIAS value or an ER group name.
password	Specifies the secondary server password	The password is used only during the first connection attempt. After the primary and secondary server have connected, the password cannot be changed.
RSS	Sets an RS secondary server type	This command should be run on the secondary database server.
pri_ha_alias	Identifies the name of the primary server	
change RSS	Change an RS secondary server	This command should be run on the primary database server.
delete RSS	Removes an RS secondary server definition	This command should be run on the primary database server.
set SDS primary	Defines the server as a shared disk primary server	
ha_alias	The high-availability alias of the database server	When used with **set SDS** or **make primary**, this is the name of the server whose role is changing.
force	Used to force a change	If the **force** option is specified, the operation is performed without requiring that the secondary server be connected to the current primary server. If the **force** option is not specified, the operation must be coordinated with the current primary server. The **force** option should be used only when the DBA is certain that the current primary server is not active; otherwise, the shared disk subsystem can become corrupted.
clear SDS primary	Disables the shared disk environment. The server name specified no longer acts as an SD primary server	

Element	Purpose	Key Considerations
make primary	Creates a primary server	The **make primary** command can be issued on any type of secondary server, including HDR secondary, RS secondary, and SD secondary servers. If **make primary** is run on: • HDR Secondary: The current primary server is shut down and the secondary is made the primary. • RS secondary: The server is changed to a standard server. • SD secondary: The server is made the new primary server.

You can also set data replication characteristics can with SQL administration API *command* equivalents. For more information see SQL Administration API Overview on page 716 and the *HCL OneDB™ Administrator's Guide*.

For other **onmode -d** information, see onmode -d: Set data-replication types on page 381 and onmode -d command: Replicate an index with data-replication on page 384.

onmode -d command: Replicate an index with data-replication

```
>>-onmode-- -d--+-idxauto--+-on--+------------------------+---->< 
                |          '-off-'                        |
                '-index--database:-+--------+-table#index-'
                                   '-owner.-'
```

Element	Purpose	Key considerations
-d	Specifies how indexes are replicated to a High-Availability Data-Replication (HDR) secondary server when an index on the secondary server becomes corrupt	You can use the onmode -d idxauto and onmode -d index commands while the server is in online mode.
idxauto	Enables automatic index replication when an index on a secondary server becomes corrupt	Use the onmode -d idxauto command to overwrite the value of the DRIDXAUTO configuration parameter within a session.
index	Replicates an index from a primary to a secondary server	If you detect a corrupt index on a secondary server, use the onmode -d index command to start replication of the index from the primary to the secondary server.
database	Specifies the database containing the index to replicate	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Administrator's Guide*.
index	Specifies the name of the index to replicate	Index must exist on table and in database specified.

Element	Purpose	Key considerations
		Syntax must conform to the Identifier segment; see the *HCL OneDB™ Administrator's Guide*.
owner	Specifies the owner of a table	You must specify the current owner of the table. Syntax must conform to the Table Name segment; see the *HCL OneDB™ Administrator's Guide*.
table	Specifies the name of the table on which the index is based	Syntax must conform to the Table Name segment; see the *HCL OneDB™ Administrator's Guide*.

The onmode -d idxauto and the onmode -d index commands provide methods to replicate an index to a secondary server containing a corrupted index. The base table will be locked during the transfer of an index. The alternative to using these options is to drop and rebuild the corrupt index on the primary server.

In the case of a fragmented index with one corrupt fragment, the onmode -d idxauto command only transfers the single affected fragment, whereas the onmode -d index command transfers the whole index.

onmode -D, -M, -Q, -S: Change decision-support parameters

```
>>-onmode--+- -D--max_priority-+------------------------------><
           +- -M--kilobytes----+
           +- -Q--queries------+
           '- -S--scans--------'
```

Element	Purpose	Key considerations
-D *max_priority*	Changes the value of MAX_PDQPRIORITY	This value must be an unsigned integer between 0 and 100. Specify *max_priority* as a factor to temper user requests for PDQ resources. For information on parameters used for controlling PDQ, see MAX_PDQPRIORITY configuration parameter on page 128 and the *HCL OneDB™ Performance Guide*.
-M *kilobytes*	Changes the value of DS_TOTAL_MEMORY	This value has a platform-dependent upper limit. If you enter a very large value and that value is too large for your platform, you will receive a message that gives you the range of values for your platform. Specify *kilobytes* for the maximum amount of memory available for parallel queries.

Element	Purpose	Key considerations
		For more information, see DS_TOTAL_MEMORY configuration parameter on page 90 and the *HCL OneDB™ Performance Guide*.
-Q *queries*	Changes the value of DS_MAX_QUERIES	This value must be an unsigned integer between 1 and 8,388,608. Specify *queries* for the maximum number of concurrently executing parallel queries. For information on parameters used for controlling PDQ, see DS_MAX_QUERIES configuration parameter on page 86 and the *HCL OneDB™ Performance Guide*.
-S *scans*	Changes the value of DS_MAX_SCANS	This value must be an unsigned integer between 10 and 1,048,576. Specify *scans* for the maximum number of concurrently executing parallel scans. For information on parameters used for controlling PDQ, see DS_MAX_SCANS configuration parameter on page 87 and the *HCL OneDB™ Performance Guide*.

These options allow you to change configuration parameters while the database server is online. The new values affect only the current instance of the database server; the values are not recorded in the ONCONFIG file. If you shut down and restart the database server, the values of the parameters revert to the values in the ONCONFIG file. For more information about these configuration parameters, see Database configuration parameters on page 3.

To check the current values for the MAX_PDQPRIORITY, DS_TOTAL_MEMORY, DS_MAX_SCANS, DS_MAX_QUERIES, and the DS_NONPDQ_QUERY_MEM configuration parameters, use onstat -g mgm. See onstat -g mgm command: Print MGM resource information on page 586.

This command has an equivalent SQL administration API function.

onmode -e: Change usage of the SQL statement cache

```
>>-onmode-- -e--mode------------------------------------------><
```

Element	Purpose	Key considerations
onmode -e `ENABLE`	Enables the SQL statement cache. For more information, see the material on improving query	User sessions use the cache only when they perform either of the following actions:

Element	Purpose	Key considerations
	performance in the *HCL OneDB™ Performance Guide*.	• Set the environment variable **STMT_CACHE** to `1` • Execute the SQL statement SET STATEMENT CACHE ON
onmode -e `FLUSH`	Flushes the statements that are not in use from the SQL statement cache	The **onstat -g ssc ref_cnt** field shows 0.
onmode -e `OFF`	Turns off the SQL statement cache	No statements are cached.
onmode -e `ON`	Turns on the SQL statement cache	All statements are cached unless the user turns it off with one of the following actions: • Set the environment variable **STMT_CACHE** to `0` • Execute the SQL statement SET STATEMENT CACHE OFF

The onmode -e changes are in effect for the current database server session only. When you restart the database server, it uses the default STMT_CACHE parameter value in the ONCONFIG file.

This command has an equivalent SQL administration API function.

onmode -F: Free unused memory segments

Use the onmode -F command to free shared-memory segments that are unavailable or no longer needed for a process

Restriction: Do not run the onmode -F command if HCL OneDB™ 11.70.xC7 is running on Solaris 11 systems. Upgrade to HCL OneDB™ 11.70.xC8 or a later version, and then run the command.

```
>>-onmode-- -F-------------------------------------------------><
```

Element	Purpose	Key considerations
-F	Frees unused memory segments	None.

When you execute onmode -F, the memory manager examines each memory pool for unused memory. When the memory manager locates blocks of unused memory, it immediately frees the memory. After the memory manager checks each memory pool, it begins checking memory segments and frees any that the database server no longer needs.

It is recommended that you run onmode -F from an operating-system scheduling facility regularly and after the database server performs any function that creates additional memory segments, including large index builds, sorts, or backups.

Running onmode -F causes a significant degradation of performance for any users that are active when you execute the utility. Although the execution time is brief (1 to 2 seconds), degradation for a single-user database server can reach 100 percent. Systems with multiple CPU virtual processors experience proportionately less degradation.

To confirm that onmode freed unused memory, check your message log. If the memory manager frees one or more segments, it displays a message that indicates how many segments and bytes of memory were freed.

This command has an equivalent SQL administration API function.

onmode -h: Update sqlhosts caches

Syntax

```
>>-onmode-- -h--+-------+--->< 
                +-force-+
```

Element	Purpose	Key considerations
-h	Allows you to update sqlhosts caches.	The database server maintains a hierarchy of sqlhosts caches. There is a cache of sqlhosts entries for each session. In addition, there exists a global cache of the sqlhosts entries which can be enabled and disabled by the use of the sqlhosts argument of the NS_CACHE configuration parameter. The global sqlhosts cache entries will be reloaded when this cache is enabled and the last modification time of the sqlhosts file is newer than the cache read time of the entry.
-force	This optional argument allows you to trigger the sqlhosts caches unconditionally.	The sqlhosts caches will be reloaded unconditionally.

This command has an equivalent SQL administration API function. For more information, see onmode and h arguments: Update sqlhosts caches (SQL administration API)

onmode -I: Control diagnostics collection

Use the **onmode -I** option to start and stop diagnostics collection.

When you encounter an error, you can specify the **onmode -I** *iserrno* option to start collecting diagnostics information. You can also specify the session ID to collect information for only specific session.

To stop the diagnostics collection, use the onmode **-I** option without any other parameters.

```
>>-onmode-- -I--+----------------------+----------------------><
                |          .-,-------. |
                |          V         | |
                '-iserrno----+-----+-+-'
                             '-sid-'
```

Element	Purpose	Key Considerations
iserrno	Message number of the error that you want to collect diagnostic information for.	None.
sid	Session ID of the session that you want to collect diagnostic information for.	None.

The diagnostics collection procedures are described in the *HCL OneDB™ Administrator's Guide*.

onmode -k, -m, -s, -u, -j: Change database server mode

```
>>-onmode--+- -k-+---------------------------------------------><
           +- -m-+
           +- -s-+
           +- -u-+
           '- -j-'
```

Element	Purpose	Key Considerations
-k	Takes the database server to offline mode and removes shared memory	To reinitialize shared memory, shut down and restart the database server. Taking the Database Server to Offline Mode with the -k Option on page 390.
-m	Takes the database server from quiescent or administration mode to online mode	See Bringing the Database Server Online with the -m Option on page 390.
-s	Shuts down the database server gracefully	Users who are using the database server are allowed to finish before the database server comes to quiescent mode, but no new connections are allowed. When all processing is finished, **-s** takes the database server to quiescent mode. The **-s** option leaves shared memory intact. See Shutting Down the Database Server Gracefully with the -s Option on page 390.
-u	Shuts down the database server immediately	This option brings the database server to quiescent mode without waiting for users to finish their sessions. Their current transactions are rolled back, and their sessions are terminated.

Element	Purpose	Key Considerations
		See Shutting Down the Database Server Immediately with the -u Option on page 390.
-j	Puts the database server into administration mode	This option brings the database server to administration mode, allowing the **informix** user all functions including the issuance of SQL and DDL commands. The **-j -U** option enables the DBSA to designate specific users (in addition to the **informix** user) to access the database server. See your *HCL OneDB™ Administrator's Guide*.

The following sections describe the options that take the database server from one mode to another.

Taking the Database Server to Offline Mode with the -k Option

The **onmode -k** option takes the database server to offline mode and removes database server shared memory.

A prompt asks for confirmation. Another prompt asks for confirmation to kill user threads before the database server comes offline. If you want to eliminate these prompts, execute the **-y** option with the **-s** option.

This option does not kill all client sessions. Use the **-u** option to avoid hanging client sessions or virtual server processes.

Important: When you use the **onmode -k** command to shut down the database server, utilities that are waiting for a user response might not terminate. For example, **onspaces** might be waiting for **y** or **n** to continue. If this problem occurs, use **onmode -uk** or **-uky** instead to roll back work before removing shared memory. For more information, see the descriptions of other options on this page.

Bringing the Database Server Online with the -m Option

The **-m** option brings the database server online from quiescent mode.

Shutting Down the Database Server Gracefully with the -s Option

The **-s** option causes a graceful shutdown. Users who are using the database server are allowed to finish before the database server comes to quiescent mode, but no new connections are allowed. When all processing is finished, **-s** takes the database server to quiescent mode. The **-s** option leaves shared memory intact.

A prompt asks for confirmation. If you want to eliminate this prompt, execute the **-y** option with the **-s** option.

Shutting Down the Database Server Immediately with the -u Option

The **-u** option causes immediate shutdown. This option brings the database server to quiescent mode without waiting for users to finish their sessions. Their current transactions are rolled back, and their sessions are terminated.

A prompt asks for confirmation. Another prompt asks for confirmation to kill user threads before the database server comes to quiescent mode. If you want to eliminate these prompts, execute the **-y** option with the **-s** option.

Changing the Database Server to Administration Mode with the -j Option

The **-j** option puts the database server into the administration mode and allows only the DBSA group and the user **informix** to connect to the server. The **-j** option allows a DBSA to have the server in a fully functional mode to perform maintenance.

The **-j -U** option enables the DBSA to grant individual users access to the database server in administration mode. Once connected, these individual users can execute any SQL or DDL command. When the server is changed to administration mode, all sessions for users other than user **informix**, the DBSA group users, and those identified in the **onmode -j -U** command lose their database server connection.

The following example enables three individual users to connect to the database server and have database server access until the database server mode changes to offline, quiescent or online mode:

```
onmode -j -U karin,sarah,andrew
```

Access for individual users can also be removed by executing **onmode -j -U** and removing their name from the new list of names in the command. For example, in the following commands, the first command grants only Karin access, the second command grants Karin and Sarah access, and the third command grants only Sarah access (and removes access from Karin).

```
onmode -j -U karin
onmode -j -U karin,sarah
onmode -j -U sarah
```

To allow user **informix** and the DBSA group user to retain their database server access in administration mode and remove all single users from accessing the database server, use the following command:

```
onmode -j -U ' '
```

For information on designating single users in administration mode using a configuration parameter, see ADMIN_MODE_USERS configuration parameter on page 33

Changing Database Server Mode with ON-Monitor (UNIX™)

You can also use ON-Monitor options to change the database server mode.

The following table shows ON-Monitor options that are equivalent to the **onmode** options.

onmode Option

ON-Monitor Option

-k

Take-Offline

-m

On-Line

-s

Graceful-Shutdown

-u

Immediate-Shutdown

-j

Administration Mode

onmode -l: Switch the logical-log file

```
>>-onmode-- -l-------------------------------------------------><
```

Element	Purpose	Key considerations
-l	Switches the current logical-log file to the next logical-log file	You must use onmode to switch to the next logical-log file. For information on switching to the next logical-log file, see the chapter on managing logical-log files in the *HCL OneDB™ Administrator's Guide*.

This command has an equivalent SQL administration API function.

onmode -n, -r: Change shared-memory residency

```
>>-onmode--+- -n-+---------------------------------------------><
           '- -r-'
```

Element	Purpose	Key considerations
-n	Ends forced residency of the resident portion of shared memory	This command does not affect the value of RESIDENT, the forced-residency parameter in the ONCONFIG file.
-r	Starts forced residency of the resident portion of shared memory	This command does not affect the value of RESIDENT, the forced-memory parameter in the ONCONFIG file.

Important: Set the RESIDENT parameter to `1` before you use the onmode -r or **-n** options.

For information on using the forced-residency parameter to turn residency on or off for the next time that you restart the database server, see the chapter on managing shared memory in the *HCL OneDB™ Administrator's Guide*.

This command has an equivalent SQL administration API function.

onmode -O: Override ONDBSPACEDOWN WAIT mode

```
>>-onmode-- -O-------------------------------------------------><
```

Element	Purpose	Key considerations
-O	Overrides the WAIT mode of the ONDBSPACEDOWN configuration parameter	None.

Use the onmode -O option only in the following circumstances:

- ONDBSPACEDOWN is set to WAIT.
- A disabling I/O error occurs that causes the database server to block all updating threads.
- You cannot or do not want to correct the problem that caused the disabling I/O error.
- You want the database server to mark the disabled dbspace as down and continue processing.

When you execute this option, the database server marks the dbspace responsible for the disabling I/O error as down, completes a checkpoint, and releases blocked threads. Then onmode prompts you with the following message:

```
This will render any dbspaces which have incurred disabling I/O errors unusable
and require them to be restored from an archive.
Do you wish to continue?(y/n)
```

If onmode does not find any disabling I/O errors on noncritical dbspaces when you run the -O option, it notifies you with the following message:

```
There have been no disabling I/O errors on any noncritical dbspaces.
```

This command has an equivalent SQL administration API function.

onmode -p: Add or drop virtual processors

Use the onmode -p command to dynamically add or drop virtual processors for the database server instance. The onmode -p command does not update the `onconfig` file.

```
>>-onmode-- -p--+-+----+--number--+-ADT-----+-+----------------><
                | '- +-'          +-AIO-----+ |
                |                 +-BTS-----+ |
                |                 +-CPU-----+ |
                |                 +-DWAVP---+ |
                |                 +-ENCRYPT-+ |
                |                 +-JVP-----+ |
                |                 +-LIO-----+ |
                |                 +-MSC-----+ |
                |                 +-PIO-----+ |
                |                 +-SOC-----+ |
                |                 +-STR-----+ |
                |                 '-vpclass-' |
                '- - -number--+-BTS-----+-----'
                              +-CPU-----+
                              +-ENCRYPT-+
                              +-JVP-----+
                              '-vpclass-'
```

Element	Purpose	Key Considerations
-p *number*	Adds or drops virtual processors. The *number* argument indicates the number of virtual processors to add or drop If this value is a negative integer, processors are dropped. If this value is a positive integer, processors are added.	You can use the **-p** option only when the database server is in online mode, and you can add to only one class of virtual processors at a time. For more details, see Rules for adding and dropping virtual processors on page 395. If you are dropping virtual processors, the maximum cannot exceed the actual number of processors of the specified type. If you are adding virtual processors, the maximum number depends on the operating system. For more information, see the chapter on using virtual processors in the *HCL OneDB™ Administrator's Guide*.
ADT	Runs auditing processes	The database server starts one virtual processor in the audit class when you turn on audit mode by setting the ADTMODE parameter in the ONCONFIG file.
AIO	Performs nonlogging disk I/O to cooked disk spaces	Also performs nonlogging I/O to raw disk spaces if kernel asynchronous I/O (KAIO) is not used.
BTS	Run basic text search index operations and queries.	BTS virtual processors are non-yielding. Specify more BTS virtual processors if you want multiple basic text search queries to be run simultaneously. Use the VPCLASS parameter in the `onconfig` file to create at least one permanent BTS virtual processor. For more information on basic text search queries, see the *HCL OneDB™ Database Extensions User's Guide*.
CPU	Runs all session threads and some system threads	It is recommended that the number of CPU VPs not be greater than the number of physical processors. If KAIO is used, performs I/O to raw disk spaces, including I/O to physical and logical logs. Runs thread for KAIO where available or a single poll thread. The database server uses the number of CPU VPs to allocate resources for parallel database queries (PDQ). If you drop CPU VPs, your queries will run significantly slower. The **Reinit** field of the **onstat -g mgm** output displays information on the number of queries that are waiting for running queries to complete after an **onmode -p** command. Also see the *HCL OneDB™ Performance Guide*.
ENCRYPT	Executes column-level encryption and decryption routines	Specify more ENCRYPT virtual processors if you have multiple encrypted columns.

Element	Purpose	Key Considerations
JVP	Executes Java™ user-defined routines in the Java™ Virtual Machine (JVM)	Specify more JVPs if you are running many Java™ UDRs.
LIO	Writes to the logical-log files if they are in cooked disk space	Use two LIO virtual processors only if the logical logs are in mirrored dbspaces. The database server allows a maximum of two LIO virtual processors.
MSC	Manages requests for system calls that require a large stack	Used for miscellaneous internal tasks.
PIO	Writes to the physical log if it is in cooked disk space	Use two PIO virtual processors only if the physical log is in a mirrored dbspace. The database server allows a maximum of two PIO virtual processors.
SOC	Uses sockets to perform network communications	You can use the SOC virtual processor only if the database server is configured for network connections through sockets.
STR	Performs stream pipe connections	
vpclass	Names a user-defined virtual processor class	Use the VPCLASS parameter in the `onconfig` to define the user-defined virtual-processor class. Specify more user-defined virtual processors if you are running many UDRs. On Windows™, you can have only one user-defined virtual processor class at a time. Omit the *number* parameter in the **onmode -p** *vpclass* command. For more information on extension classes, see VPCLASS configuration parameter on page 212.

Rules for adding and dropping virtual processors

You can add or drop virtual processors.

The following rules apply:

- You cannot drop the final virtual processor. At least one virtual processor must remain.
- You cannot add or drop ADM or OPT.
- **Windows™ Only:** You can add a supported virtual processor of any class, but you cannot drop virtual processors.

These are the virtual processors that you can add or drop:

Virtual processor name	Add	Drop
ADT	Yes	No
AIO	Yes	No
BTS	Yes	Yes
CPU	Yes	Yes
ENCRYPT	Yes	Yes
JVP	Yes	Yes
LIO	Yes[1 on page 396]	No
MSC	Yes	No
PIO	Yes[1 on page 396]	No
SOC	Yes	No
STR	Yes	No
vpclass	Yes	Yes

Table note:

1. You can add one more virtual processor.

Monitoring poll threads with the onstat utility

About this task

While the database server is online, you cannot drop a CPU virtual processor that is running a poll thread. To identify poll threads that run on CPU virtual processors, use the following command:

```
onstat -g ath | grep 'cpu.*poll'
```

The following **onstat -g ath** output shows two CPU virtual processors with poll threads. In this situation, you cannot drop to fewer than two CPU virtual processors.

```
tid  tcb      rstcb  prty  status    vp-class  name

8     a362b90  0      2     running   1cpu      tlitcppoll
9     a36e8e0  0      2     cond wait arrived   3cpu
```

The `status` field contains information, such as `running`, `cond wait`, `IO Idle`, `IO Idle`, `sleeping secs: number_of_seconds`, or `sleeping forever`. To improve performance, you can remove or reduce the number of threads that are identified as `sleeping forever`.

For more information on the types of virtual processors, see the chapter on virtual processors and threads in the *HCL OneDB™ Administrator's Guide*.

This command has an equivalent SQL administration API function.

onmode -P: Start, stop, or restart a listen thread dynamically

Use the onmode -P command to start, stop, or restart an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.

```
>>-onmode-- -P--+-start---+--server_name----------------------><
                +-stop----+
                '-restart-'
```

Element	Purpose	Key Considerations
start	Start a new listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.	The definition of the listen thread must exist in the **sqlhosts** file for the server. If the definition of the listen thread does not exist in the **sqlhosts** file, you must add it before you can start the listen thread dynamically.
stop	Stop an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.	The definition of the listen thread must exist in the **sqlhosts** file for the server.
restart	Stop and start an existing listen thread for a SOCTCP or TLITCP network protocol without interrupting existing connections.	The definition of the listen thread must exist in the **sqlhosts** file for the server.
server_name	The name of the database server on which you want to start, stop, or restart a listen thread.	

These commands do not update the **sqlhosts** file.

These commands are equivalent to the SQL administration API functions that have **start listen**, **stop listen**, or **restart listen** arguments.

Example

Example

The following command stops and then starts a listen thread for a server named **ids_serv1**:

```
onmode -P restart ids_serv1
```

onmode -R: Regenerate .infos.dbservername File

The database server creates the `.infos.dbservername` file when you initialize shared memory and removes the file when you take the database server offline. This file is in the $ONEDB_HOME/etc or %ONEDB_HOME%\etc directory. The name of this file is derived from the DBSERVERNAME parameter in the ONCONFIG configuration file.

The database server uses information from the `.infos.dbservername` file when it accesses utilities. If the file is accidentally deleted, you must either re-create the file or shut down and restart the database server.

```
>>-onmode-- -R-------------------------------------------------><
```

Element	Purpose	Key Considerations
-R	Re-creates the `.infos.dbservername` file	Before you use the **-R** option, set the ONEDB_SERVER environment variable to match the DBSERVERNAME parameter from the ONCONFIG file. Do not use the **-R** option if the ONEDB_SERVER environment variable is set to one of the DBSERVERALIASES names.

onmode -W: Change settings for the SQL statement cache

```
>>-onmode-- -W--+-STMT_CACHE_HITS--hits-----+------------------><
                '-STMT_CACHE_NOLIMIT--value-'
```

Element	Purpose	Key Considerations
STMT_CACHE_HITS *hits*	Specifies the number of hits (references) to a statement before it is fully inserted in the SQL statement cache Set ***hits*** to 1 or more to exclude ad hoc queries from entering the cache.	You can only increase or reset the value of STMT_CACHE_HITS. The new value displays in the **#hits** field of the **onstat -g ssc** output. If ***hits*** = 0, the database server inserts all qualified statements and its memory structures in the cache. If ***hits*** > 0 and the number of times the SQL statement has been executed is less than STMT_CACHE_HITS, the database server inserts *key-only* entries in the cache. It inserts qualified statements in the cache after the specified number of hits have been made to the statement. **ONCONFIG** Parameter: STMT_CACHE_HITS
STMT_CACHE_NOLIMIT *value*	Controls whether statements are inserted in the SQL statement cache.	If *value* = 0, the database server inserts no statements in the cache. If *value* = 1, the database server always inserts statements in the cache. If none of the queries are shared,

Element	Purpose	Key Considerations
		turn off STMT_CACHE_NOLIMIT to prevent the database server from allocating a large amount of memory for the cache.**ONCONFIG** Parameter: STMT_CACHE_NOLIMIT

SQL statement cache examples

The following are examples of onmode -W commands for changing SQL statement cache (SSC) settings. The changes are in effect for the current database server session only and do not change the ONCONFIG values. When you restart the database server, it uses the default SSC settings, if not specified in the ONCONFIG file, or the ONCONFIG settings. To make the changes permanent, set the appropriate configuration parameter.

```
onmode -W STMT_CACHE_HITS 2  # number of hits before statement is
     # inserted into SSC
onmode -W STMT_CACHE_NOLIMIT 1  # always insert statements into
     # the cache
```

This command has an equivalent SQL administration API function.

onmode -we: Export a file that contains current configuration parameters

Use the onmode -we command to create and export a configuration file that is a snapshot of your current configuration parameters.

```
>>-onmode-- -we--path_name-------------------------------------><
```

Element	Description	Key Considerations
path_name	The full or relative path name of the configuration file.	Do not add an extension.

Usage

The onmode -we command automatically creates an ASCII file, assigning it the name that you specified in the command. The format of the file is the same as the format of the `onconfig.std` file.

If you changed any values dynamically during the current session, the exported file contains the changed values instead of the values that are permanently saved in the `onconfig` file.

After you export the configuration file, you can import it and use it as your configuration file.

If run the onmode -we command and specify a file that was previously exported, the command exports the new version of the file, overwriting the previous exported file.

The onmode -we command is equivalent to the SQL administration API function that has the **onmode** and **export** arguments.

Example

Examples

The following command exports all configuration parameters and their current values to the `onconfig3` file in the `/tmp` directory:

```
onmode -we /tmp/onconfig3
```

onmode -wf, -wm: Dynamically change certain configuration parameters

Use the onmode -wf or onmode -wm command to dynamically change specific configuration parameters.

```
>>-onmode--+- -wf--config_param=value-+------------------------><
           '- -wm--config_param=value-'
```

Element	Purpose	Key considerations
-wf	Updates the value of the specified configuration parameter in the `onconfig` file.	The DBA user must have write permission for the directory that contains the `onconfig` file.
-wm	Dynamically sets the value of the specified configuration parameter in memory.	The specified *value* is not preserved when the server is restarted.
config_param=value	Specifies the configuration parameter and its new value.	See Database configuration parameters on page 3.

To see a list of configuration parameters that you can tune dynamically with an onmode -wm or -wf command, run the onstat -g cfg tunable command.

You can use onmode -wm or onmode -wf to change the following configuration parameters:

- ADMIN_MODE_USERS
- ALARMPROGRAM
- AUTO_AIOVPS
- AUTO_CKPTS
- AUTO_LRU_TUNING
- AUTO_READAHEAD
- AUTO_REPREPARE
- AUTO_STAT_MODE
- AUTOLOCATE
- BATCHEDREAD_TABLE
- BATCHEDREAD_INDEX
- BLOCKTIMEOUT

- CDR_DELAY_PURGE_DTC
- CDR_LOG_LAG_ACTION
- CDR_LOG_STAGING_MAXSIZE
- CKPTINTVL
- DBSPACETEMP
- DEADLOCK_TIMEOUT
- DEF_TABLE_LOCKMODE
- DELAY_APPLY
- DIRECTIVES
- DRINTERVAL
- DRTIMEOUT
- DS_MAX_QUERIES
- DS_MAX_SCANS
- DS_NONPDQ_QUERY_MEM
- DS_TOTAL_MEMORY
- DUMPCNT
- DUMPSHMEM
- DYNAMIC_LOGS
- ENABLE_SNAPSHOT
- EXPLAIN_STAT
- FILLFACTOR
- HA_ALIAS
- IFX_EXTEND_ROLE
- IFX_FOLDVIEW
- CONNECT_RETRIES
- CONNECT_TIMEOUT
- LIMITNUMSESSIONS
- LISTEN_TIMEOUT
- LOG_INDEX_BUILDS
- LOG_STAGING_DIR
- LOGSIZE
- LOW_MEMORY_MGR (onmode -wf only, not onmode -wm)
- LOW_MEMORY_RESERVE
- LTAPEBLK
- LTAPEDEV
- LTAPESIZE
- LTXEHWM
- LTXHWM
- MAX_INCOMPLETE_CONNECTIONS
- MAX_PDQPRIORITY
- MSG_DATE
- MSGPATH

- NET_IO_TIMEOUT_ALARM
- NS_CACHE
- ONDBSPACEDOWN
- ONLIDX_MAXMEM
- OPTCOMPIND
- REMOTE_SERVER_CFG
- REMOTE_USERS_CFG
- RESIDENT
- RSS_FLOW_CONTROL
- RTO_SERVER_RESTART
- S6_USE_REMOTE_SERVER_CFG
- SBSPACENAME
- SBSPACETEMP
- SDS_TIMEOUT
- SHMADD
- SMX_COMPRESS
- SP_AUTOEXPAND
- SP_THRESHOLD
- SP_WAITTIME
- SQL_LOGICAL_CHAR
- STACKSIZE
- STATCHANGE
- STOP_APPLY
- SYSALARMPROGRAM
- SYSSBSPACENAME
- TAPEBLK
- TAPEDEV
- TAPESIZE
- TBLTBLFIRST
- TBLTBLNEXT
- TEMPTAB_NOLOG
- TXTIMEOUT
- USELASTCOMMITTED
- USTLOW_SAMPLE
- VP_MEMORY_CACHE_KB
- WSTATS

The onmode -wf and onmode -wm commands have equivalent SQL administration API functions.

onmode -wm: Change LRU tuning status

You can use the onmode -wm option to change the LRU tuning status without updating the `onconfig` file.

```
>>-onmode-- -wm--AUTO_LRU_TUNING----+-0-+---------------------><
                                    '-1-'
```

Element	Purpose	Key considerations
-wm	Dynamically sets the value of the specified configuration parameter for the current session.	None.
`0`	Turns off automatic LRU tuning for the current session.	None.
`1`	Turns on automatic LRU tuning for the current session.	None.

This command has an equivalent SQL administration API function.

onmode -wi: Import a configuration parameter file

Use the onmode -wi command to import a file that contains new values for multiple configuration parameters. If the parameters are tunable, which means they can be updated individually with an onmode -wm command, the database server applies the new values.

```
>>-onmode-- -wi--path_name------------------------------------><
```

Element	Purpose	Key Considerations
path_name	The full or relative path name of the previously exported configuration file.	

Usage

Importing a configuration file with onmode -wi is often faster and more convenient than running individual onmode -wm commands on multiple tunable configuration parameters.

The import operation ignores the configuration parameters in the file that are not tunable. The operation also ignores new parameter values that match the values that are currently used by the instance.

After you import the file, you can modify the values of the imported configuration parameters.

An import operation changes only the values of configuration parameters that are in memory. The operation does not affect the values in the `$ONEDB_HOME/etc/$ONCONFIG` file.

The onmode -wi command is equivalent to the SQL administration API functions that have **onmode** and **wi** arguments or the **import** argument.

Example

Example

The following command imports the configuration parameters that are in a file named `onconfig3` in the `/tmp` directory:

```
onmode -wi /tmp/onconfig3
```

onmode -Y: Dynamically change SET EXPLAIN

```
>>-onmode-- -Y--session_id--+-0--------------------+-----------><
                            '-+-2-+--+-----------+-'
                              '-1-'  '-file_name-'
```

Element	Purpose	Key considerations
file_name	The explain output file name.	If the file's absolute path is not included, the example output file is created in the default example output file location. If the file exists, explain output is appended to it. If a file exists from the SET EXPLAIN statement, that file is not used until dynamic explain is turned off.
session_id	Identifies the specific session.	None.
-Y	Dynamically change the value of the SET EXPLAIN statement.	None.

You can use the SET EXPLAIN statement to display the query plan of the optimizer, an estimate of the number of rows returned, and the relative cost of the query. When you use the onmode -Y command to turn on SET EXPLAIN, the output is displayed in the explain output file.

The onmode -Y command dynamically changes the value of the SET EXPLAIN statement for an individual session. The following invocations are valid with this command:

Invocation	Explanation
`onmode -Y session_id 2`	Turns SET EXPLAIN on for *session_id*
`onmode -Y session_id 1`	Turns SET EXPLAIN on for *session_id* and displays the query statistics section in the explain output file
`onmode -Y session_id 1 /tmp/myexplain.out`	Turns SET EXPLAIN on for *session_id* and writes explain output to `/tmp/myexplain.out`.
`onmode -Y session_id 0`	Turns SET EXPLAIN off for *session_id*

This command has an equivalent SQL administration API function.

onmode -z: Kill a database server session

```
>>-onmode-- -z--sid--------------------------------------------><
```

Element	Purpose	Key considerations
-z *sid*	Kills the session that you specify in *sid*	This value must be an unsigned integer greater than 0 and must be the session identification number of a currently running session.

To use the **-z** option, first obtain the session identification (*sessid*) with onstat -u, then execute onmode -z, substituting the session identification number for *sid*.

When you use onmode -z, the database server attempts to kill the specified session. If the database server is successful, it frees any resources that the session holds. If the database server cannot free the resources, it does not kill the session.

If the session does not exit the section or release the latch, the database server administrator can take the database server offline, as described in Taking the Database Server to Offline Mode with the -k Option on page 390, to close all sessions.

This command has an equivalent SQL administration API function.

onmode -Z: Kill a distributed transaction

```
>>-onmode-- -Z--address---------------------------------------><
```

Element	Purpose	Key considerations
-Z *address*	Kills a distributed transaction associated with the shared-memory address *address*	This argument must be the address of an ongoing distributed transaction that has exceeded the amount of time that TXTIMEOUT specifies. The address must conform to the operating-system-specific rules for addressing shared-memory. (The address is available from onstat -x output.) This option is not valid until the amount of time that the ONCONFIG parameter TXTIMEOUT specifies has been exceeded. The **-Z** option should rarely be used and only by an administrator of a database server involved in distributed transactions. For information on initiating independent actions in a two-phase commit protocol, see the chapter on multiphase commit protocols in the *HCL OneDB™ Administrator's Guide*.

Distributed transactions provide the ability to query data on different database servers.

Attention: If applications are performing distributed transactions, killing one of the distributed transactions can leave your client/server database system in an inconsistent state. Try to avoid this situation.

This command has an equivalent SQL administration API function.

The ON-Monitor Utility

Using ON-Monitor (UNIX™)

Use the ON-Monitor utility to perform various administrative tasks.

This section provides a quick reference for the ON-Monitor screens. To start ON-Monitor, execute the following command from the operating-system prompt:

Example

```
onmonitor
```

If you are logged in as user **informix** or user **root**, the main menu appears. All users other than **informix** and **root** have access only to the Status menu.

The ON-Monitor main menu displays the following menus:

- **Status** menu
- **Parameters** menu
- **Dbspaces** menu
- **Mode** menu
- **Force-Ckpt** menu
- **Archive** menu
- **Logical-Logs** menu
- **Exit** option

These menus are shown on the following pages (Table 125: Status Menu on page 407 through Table 131: Logical Logs Menu on page 409).

To obtain ON-Monitor version information, execute the **-V** or **-version** command from the operating-system prompt. For complete information on version information, see Obtaining utility version information on page 312

You cannot use ON-Monitor on High-Availability Data Replication (HDR) secondary servers, remote standalone (RS) secondary servers, or shared disk (SD) secondary servers.

Navigating ON-Monitor and Using Help

All menus and screens in ON-Monitor function in the same way.

For menus, use the arrow keys or `SPACEBAR` to scroll to the option that you want to execute and press `RETURN`, or press the first capitalized letter of the option (usually the first letter). When you move from one option to the next by pressing `SPACEBAR` or an arrow key, the option explanation (line 2 of the menu) changes.

If you want general instructions for a specific screen, press `CTRL-W`. If you need help to determine what you should enter in a field on the screen, use the `TAB` key to highlight the field and press `CTRL-F` or `F2`.

Some of the menus display ellipses (...) on the far right or left side. The ellipses indicate that you can move in the direction of the dots, using the arrow keys or `SPACEBAR`, to view other options.

Executing Shell Commands Within ON-Monitor

To execute a shell command from within ON-Monitor, type an exclamation point (`!`) followed by the command.

About this task

For example, to list the files in the current directory, type the following command:

```
!ls
```

ON-Monitor Screen Options

This topic describes the options on ON-Monitor menus.

Table 125. Status Menu

Menu	Description
Profile	Displays database server performance statistics
Userthreads	Displays the status of active user threads
Spaces	Displays status information about database server storage spaces and chunks
Databases	Displays the name, owner, and logging mode of the 100 first databases
Logs	Displays status information about the physical-log buffer, the physical log, the logical-log buffer, and the logical-log files
data-Replication	Displays High-Availability Data-Replication (HDR) status and configuration
Output	Stores the output of other status information in a specified file
Configuration	Copies the current database server configuration to a file

Table 126. Parameters Menu

Menu	Description
Initialize	Initializes database server disk space or modifies disk-space parameters
Shared-Memory	Initializes database server shared memory or modifies shared-memory parameters
perFormance	Specifies the number of virtual processors for each VP
data-Replication	Specifies the HDR parameters
diaGnostics	Specifies values for the diagnostics parameters
pdQ	Changes parameters for parallel database queries
Add-Log	Adds a logical-log file to a dbspace
Drop-Log	Drops a logical-log file from a dbspace
Physical-Log	Changes the size or the location of the database server physical log

Table 127. Dbspaces Menu

Menu	Description
Create	Creates a dbspace
BLOBSpace	Creates a blobspace
Mirror	Adds mirroring to an existing storage space or ends mirroring for a storage space
Drop	Drops a storage space from the database server configuration
Info	Displays the identification number, location, and fullness of each chunk assigned to a storage space
Add_chunk	Adds a chunk to a storage space
datasKip	Changes the database parameter
Status	Changes the status of a chunk in a mirrored pair

Table 128. Mode Menu

Menu	Description
Startup	Initializes shared memory and takes the database server to quiescent mode
On-Line	Takes the database server from quiescent to online mode
Graceful-Shutdown	Takes the database server from online to quiescent mode so users can complete work
Immediate-Shutdown	Takes the database server from online to quiescent mode in 10 seconds
Take-Offline	Detaches shared memory and immediately takes the database server to offline mode
Add-Proc	Adds virtual processors
Drop-Proc	Drops virtual processors
Decision-support	Sets decision-support parameters dynamically
Administration	Tells the server to change into administration mode

Table 129. Force-Ckpt Menu

Menu	Description
Force-Ckpt	Displays the time of the most-recent checkpoint or forces the database server to execute a checkpoint

Table 130. Archive Menu

Menu	Description
Tape-Parameters	Modifies the **ontape** parameters for the backup tape device

Table 131. Logical Logs Menu

Menu	Description
Databases	Modifies the logging status of a database
Tape-Parameters	Modifies the **ontape** parameters for the logical-log backup tape device

The onparams Utility

Use the **onparams** utility to add or drop a logical-log file, change physical-log parameters, and add a new buffer pool.

In This Chapter

This chapter shows you how to use the following **onparams** options:

- onparams -a -d dbspace: Add a logical-log file on page 410
- onparams -d -l lognum: Drop a logical-log file on page 410
- onparams -p: Change physical-log parameters on page 411
- onparams -b: Add a buffer pool on page 412

Any **onparams** command fails if a storage-space backup is in progress. If you do not use any options, **onparams** returns a usage statement.

You cannot use the **onparams** utility on High-Availability Data Replication (HDR) secondary servers, remote standalone (RS) secondary servers, or shared disk (SD) secondary servers.

You can also use SQL administration API commands that are equivalent to **onparams** commands to add or drop a logical-log file, change physical-log parameters, and add a new buffer pool.

On UNIX™, you must be logged in as user **root** or user **informix** to execute **onparams**. Only user **informix** is allowed to execute the SQL administration API *command* strings.

On Windows™, you must be a member of the **Informix-Admin** group to execute **onparams**.

onparams syntax

Use the onparams utility to modify the configuration of logical logs or physical logs.

```
>>-onparams--+-------------------------+--+-----------------+--><
             |                     (1) |  +- -a -d--dbspace-+
             '-|  -FILE option   |-----'  +- -d -l--lognum--+
                                          +- -p-------------+
                                          +- -b-------------+
                                          +- -V-------------+
                                          '- -version-------'
```

Element	Purpose	Key Considerations
-**V**	Displays the software version number and the serial number	See Obtaining utility version information on page 312.
-**version**	Displays the build version, host, OS, number and date, as well as the GLS version	See Obtaining utility version information on page 312.

onparams -a -d *dbspace*: Add a logical-log file

```
>>-onparams-- -a-- -d--dbspace--+-----------+--+-----+---------><
                                '- -s--size-'  '- -i-'
```

Element	Purpose	Key considerations
-a -d *dbspace*	Adds a logical-log file to the end of the log-file list to the specified *dbspace*	You can add a log file to a dbspace only if the database server has adequate contiguous space. The newly added log files have a status of **A** and are immediately available for use. You can add a log file during a backup. You can have a maximum of 32,767 logical-log files. Use onstat -l to view the status of your logical-log files. It is recommended that you take a level-0 backup of the root dbspace and the dbspace that contains the log file as soon as possible. You cannot add a log file to a blobspace or sbspace. Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*.
-i	Inserts the logical-log file after the current log file	Use this option when the Log File Required alarm prompts you to add a logical-log file.
-s *size*	Specifies a size in kilobytes for the new logical-log file	This value must be an unsigned integer greater than or equal to 200 kilobytes If you do not specify a size with the **-s** option, the size of the log file is taken from the value of the LOGSIZE parameter in the ONCONFIG file when database server disk space was initialized. For information on changing LOGSIZE, see the chapter on managing logical-log files in the *HCL OneDB™ Administrator's Guide*.

This command has an equivalent SQL administration API function.

onparams -d -l *lognum*: Drop a logical-log file

```
>>-onparams-- -d-- -l--lognum--+-----+------------------------><
                               '- -y-'
```

Element	Purpose	Key considerations
-d -l *lognum*	Allows you to drop a logical-log file specified by the log file number	**Restrictions**: The *lognum* value must be an unsigned integer greater than or equal to 0. You can obtain the ***lognum*** from the **number** field of onstat -l. The sequence of ***lognum*** might be out of order.
-y	Causes the database server to automatically respond yes to all prompts	None.

Usage

You can only drop one log files at a time.

The database server requires a minimum of three logical-log files at all times. You cannot drop a log if your logical log is composed of only three log files.

Important: After you add the new logical-log files and run a backup, you can use onparams -d -l*lognum* to delete the first three logical-log files.

The status of the log file determines if the log file can be dropped, and the actions taken by the database server when the log file is dropped:

- If you drop a log file that has never been written to, status is newly Added (**A**), the database server deletes the log file and frees the space immediately.
- If you drop a used log file that has a status of User (**U**) or Free (**F**), the database server marks the log file as Deleted (**D**). After you take a level-0 backup of the dbspaces that contain the log files and the root dbspace, the database server deletes the log file and frees the space.
- You cannot drop a log file that is currently in use (**C**) or contains the last checkpoint record (**L**).

This command has an equivalent SQL administration API function.

When you move logical-log files to another dbspace, use the onparams commands to add and drop logical-log files. See moving a logical-log file, in the section on managing logical-log files in the *HCL OneDB™ Administrator's Guide*.

onparams -p: Change physical-log parameters

```
>>-onparams-- -p-- -s--size--+--------------+--+-----+---------><
                             '- -d--dbspace-'  '- -y-'
```

Element	Purpose	Key Considerations
-p	Changes the physical log	Whenever you use the onparams -p command, you must include the -s parameter. Additionally, you can specify the -d and -y

Element	Purpose	Key Considerations
		parameters. The database server must be in either administration, online, or quiescent mode to specify the -p parameter. The database server does not need to be restarted for the changes take effect.
-s *size*	Changes the size (in kilobytes) of the physical log	This value must be an unsigned integer greater than or equal to 200 kilobytes. **Attention:** If you move the log to a dbspace without adequate contiguous space or increase the log size beyond the available contiguous space, the operation will fail and the physical log will not change.
-d *dbspace*	Changes the location of the physical log to the specified *dbspace*	The space allocated for the physical log must be contiguous. Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*.
-y	Causes the database server to automatically respond yes to all prompts	None.

Backing Up After You Change the Physical-Log Size or Location

About this task

The database server must be in either the online or quiescent mode when you change the physical log. The database server does not need to be restarted for the changes to take effect.

Create a level-0 backup of the root dbspace immediately after you change the physical-log size or location. This backup is critical for proper recovery of the database server.

Changing the Size of the Physical Log and Using Non-Default Page Sizes

If you use non-default page sizes, you might need to increase the size of your physical log. If you perform many updates to non-default pages you might need a 150 to 200 percent increase of the physical log size. Some experimentation might be needed to tune the physical log. You can adjust the size of the physical log as necessary according to how frequently the filling of the physical log triggers checkpoints.

onparams -b: Add a buffer pool

Use the onparams -b command to create a buffer pool that corresponds to the page size of the dbspace.

```
>>-onparams-- -b-- -g--size------------------------------------><
```

Element	Purpose	Key considerations
-b	Creates a buffer pool	You can add a buffer pool while the database server is running.
-g *size*	Specifies the size in KB of the buffer pages to create	The size of the buffer pages must be 2 - 16 KB and a multiple of the default page size.

Element	Purpose	Key considerations
-b	Creates a new buffer pool	You can add a new buffer pool while the database server is running.
-g *size*	Specifies the size in kilobytes of the buffer pages to create	Each dbspace you create with a non-default page size must have a corresponding buffer pool with the corresponding page size. If you create a dbspace with a page size that has no buffer pool, the system will automatically create a buffer pool using the fields in the default line of the BUFFERPOOL parameter. The size of the buffer pages must be between 2 and 16 kilobytes and it must be a multiple of the default page size.
-m *percent*	Specifies the percentage of modified pages in the LRU queues at which page cleaning is no longer mandatory	Fractional values are allowed. If you do not specify this option, the percentage used is the value of the *lru_min_dirty* field as set in the default line of the BUFFERPOOL configuration parameter.
-n *number*	Specifies the number of buffers in the buffer pool	The range is 500 - 2147483647. If you do not specify this option, the number used is the value of *buffers* as set in the default line of the BUFFERPOOL configuration parameter.
-r *number*	Specifies the number of LRU (least-recently-used) queues in the shared-memory buffer pool	Range of values: • 32-bit platforms: 1 - 128 • 64-bit platforms: 1 - 512 If you do not include this option, the number of LRU queues allocated is equal to the value of *lrus* as set in the default line of the BUFFERPOOL configuration parameter.
-x *percent*	Specifies the default percentage of modified pages in the LRU queues at which the queue is cleaned	Fractional values are allowed. If you do not specify this option, the percentage used is the value of *lru_max_dirty* as set in the default line of the BUFFERPOOL configuration parameter.

The values of the elements correspond to the values of the fields of the BUFFERPOOL configuration parameter.

All other characteristics of the buffer pool that you create are set to the values of the fields in the default line of the BUFFERPOOL configuration parameter.

Each dbspace that you create with a non-default page size must have a corresponding buffer pool with the corresponding page size. If you create a dbspace with a page size that has no buffer pool, the system automatically creates a buffer pool based the fields in the default line of the BUFFERPOOL parameter.

You should create a buffer pool for a dbspace before you create the dbspace. You cannot reduce or increase the number of buffers in an existing buffer pool while the database server is running. You also cannot drop a buffer pool while the database server is running. You can, however, add new buffer pools with a new size while the database server is running.

Buffer pools added with the onparams utility are put into virtual memory, not into resident memory. Upon restart, buffer pool entries will go into resident memory depending on the amount of memory that is available.

When you add a buffer pool, a new entry for the BUFFERPOOL configuration parameter is added in the `onconfig` file.

This command has an equivalent SQL administration API function.

Examples of onparams Commands

The following are examples of **onparams** commands:

```
onparams -a -d rootdbs -s 1000  # adds a 1000-KB log file to rootdbs
```

```
onparams -a -d rootdbs -i       # inserts the log file after the current log
```

```
onparams -d -l 7                # drops log 7
```

```
onparams -p -d dbspace1 -s 3000 # resizes and moves physical-log to dbspace1
```

```
onparams -b -g 6 -n 3000 -r 2 -x 2.0 -m 1.0 # adds 3000 buffers of size
6K bytes each with 2 LRUS with maximum dirty of 2% and minimum dirty of 1%
```

The onpassword utility

Use the onpassword utility to encrypt and decrypt a password file. Connection Manager and Enterprise Replication utilities require a password file to connect to database servers over an untrusted network.

Syntax

```
>>-onpassword-- -k --encryption_key----------------------------->

>--+- -e --text_file--------+---------------------------------><
   '- -d --output_file_name-'
```

Element	Purpose	Key Considerations
-k	Specifies the password key.	
-e	Encrypts an ASCII text file	The password information is encrypted to `$ONEDB_HOME/etc/passwd_file`

Element	Purpose	Key Considerations
-d	Decrypts the specified encrypted password file.	The `passwd_file` is decrypted to *`$ONEDB_HOME/etc/output_file_name`*.
output_file_name	The name of the file that is output by the decryption process.	An encrypted password file that is created on one type of platform is not supported on a different type of platform. You must run the onpassword utility on each type of platform, and use the same text file and encryption key.
encryption_key	The encryption key used to encrypt and decrypt password information.	The encryption key can be any sequence of numbers or letters up to 24 bytes in length. To use an encryption key that includes spaces, enclose the encryption key in quotation marks. For example: `"my secret encryption key"`
text_file	The ASCII text file that contains user password information.	The onpassword utility uses the following default location: • UNIX™: `$ONEDB_HOME/tmp` • Windows™: `%ONEDB_HOME%\etc`

Usage

Only users logged in as user **informix** have permission to run the **onpassword** utility.

Example

Example 1: Encrypting a password file

To encrypt `tmp/my_passwords.txt` with **my_secret_encryption_key**, run the following command:

```
onpassword -k my_secret_encryption_key -e my_passwords.txt
```

The password information is encrypted into `$ONEDB_HOME/etc/passwd_file`.

Example

Example 2: Decrypting an encrypted password file

To decrypt `$ONEDB_HOME/etc/passwd_file` with **my_secret_encryption_key**, run the following command:

```
onpassword -k my_secret_encryption_key -d my_passwords.txt
```

The password information is decrypted to `$ONEDB_HOME/etc/my_passwords.txt`.

The ifxclone utility

You use the ifxclone utility to create a server clone from a snapshot of an existing database server.

Syntax

```
>>-ifxclone--+-| Mandatory parameters |--+-------------------------+-+-><
             |                            '-| Optional parameters |-' |
             '- help -----------------------------------------------'

Mandatory parameters

|-- --source=source_name --sourceIP=source_IP --sourcePort=source_port-->

>-- --target=target_name --targetIP=target_IP --targetPort=target_port--|

Optional parameters

   .----------------------------------.
   V                                  |
|----+-----------------------------+-+--+------------+-------->
     '- --configparmparameter=value-'    '- --autoconf-'

>--+------------------------+--+-------------+----------------->
   '- --disposition=-+-ER--+-'  '- --useLocal-'
                     +-HDR-+
                     +-RSS-+
                     '-SDS-'

>--+--------------+--+------------+--+--------------------+-----|
   '- --size=size-'  '- --trusted-'  '- --createchunkfile-'
```

Element	Purpose	Key Considerations
disposition	Specifies the final disposition of the new server instance.	If the --disposition (-d) parameter is not specified, a standard server is created.
ER	Specifies that the new server instance is created as a replication server.	
HDR	Specifies that the new server instance is created as an HDR secondary server.	
parameter=value	Specifies an optional configuration parameter and value to set on the target server.	Certain configuration parameters on the source server must match those on the target server. See Prerequisites for cloning an RS secondary server on page 422.
RSS	Specifies that the new server instance is created as an remote stand-alone secondary server.	
SDS	Specifies that the new server instance is created as a shared-disk secondary server.	The ifxclone utility sets the target server's SDS_PAGING, and SDS_TEMPDBS configuration parameters, but full configuration is outside the scope of the ifxclone utility.

Element	Purpose	Key Considerations
		If --disposition=SDS is specified in the command, but --useLocal is not, you must set the SD secondary server's ROOTPATH configuration parameter to the same value as the ROOTPATH configuration parameter on the primary server.
size	Specifies the size of the target server. Valid values are `tiny`, `small`, `medium`, and `large`.	If the size parameter is not specified, the size parameters from the source instance are used.
source_name	Specifies the name of the source instance.	The source server must be a primary server and cannot be a secondary server.
source_IP	Specifies the source server instance TCP/IP address.	
source_port	Specifies the TCP/IP port address of the source server instance, or the name of a service associated with the port.	
target_name	Specifies the name of the target server instance.	
target_IP	Specifies the target server instance TCP/IP address.	
target_port	Specifies the TCP/IP port address of the target server instance, or the name of a service associated with the port.	

The following table describes the options of the ifxclone utility.

Long Form	Short Form	Meaning
--autoconf	-a	Autoconfigures connectivity information between a newly cloned server and the other servers of a high-availability cluster or Enterprise Replication domain. If this option is used to create a replication server, the --autoconf option can autoconfigure replication. The --autoconf option has the following requirements:

Long Form	Short Form	Meaning
		• The CDR_AUTO_DISCOVER configuration parameter must be set to `1` on the target server, the source server, and all other cluster or replication servers. • REMOTE_SERVER_CFG must be set on all cluster or replication servers. • The target server's host information must be in the source server's trusted-host file. • If used with the --disposition=ER option, and the primary server is part of an Enterprise Replication, all other replication servers in the domain must be active.
--configParm	-c	Specifies the name and value of a configuration parameter to set on the target server.
--createchunkfile	-k	Automatically creates the same cooked chunk files on the target server as exist on the source server. This option does not create raw chunks. However, if you have raw chunks on the source server and you do not create matching raw chunks on the target server before using this option, the ifxclone utility creates cooked chunks on the target server that match the raw chunks on the source server.
--disposition	-d	Specifies the disposition of the new server instance.
--help	-h	Displays usage information.
--size	-s	Specifies the size of the target instance.
--source	-S	Specifies the name of the source server instance.
--sourceIP	-I	Specifies the TCP/IP address of the source server instance.
--sourcePort	-P	Specifies the TCP/IP port address of the source server instance, or the name of a service associated with the port.
--target	-t	Specifies the name of the target server instance.
--targetIP	-i	Specifies the TCP/IP address of the target server instance.
--targetPort	-p	Specifies the TCP/IP port address of the target server instance, or the name of a service associated with the port.
--trusted	-T	Specifies that the server is trusted and that it is not necessary to obtain a userid and password to access the server.
--useLocal	-L	Specifies that the configuration information contained in the source server `onconfig` file should be merged with the target server `onconfig` file.

Long Form	Short Form	Meaning
		Certain configuration parameters on the source server must match those on the target server. See Prerequisites for all servers on page 420.

Usage

Use the ifxclone utility to clone a server with minimum setup or configuration, or to quickly add a new node to an existing ER replication domain. When the ifxclone utility is run, the majority of the setup information is obtained from the server node that is being cloned. Successfully cloning a server might still require some post-configuration steps to achieve a better running system.

The source server is the server that contains the data you wish to clone. The target server is the server that is to be loaded with data from the source server. You must run the ifxclone utility from the target server.

To run the ifxclone utility on a UNIX™ computer, you must run the command on the target server as user **root**, user **informix**, or as a member of the **informix** group. You must also be a DBSA on the source server.

To run the ifxclone utility on a Windows™ computer, you must run the command on the target server as a member of the local administrators group. You must also be a DBSA on the source server and you must belong to the **Informix-Admin** group on the source server.

The ifxclone utility uses the `onconfig` and `sqlhosts` configuration files from the source server to configure the target server. The ifxclone utility also configures some additional configuration settings, but only those required to configure the clone server. The --autoconf option provides the additional ability to configure `sqlhosts` file records, and then propagate `sqlhosts` and trusted-host file information to the servers of a high-availability cluster or Enterprise Replication domain. The --createchunkfile options creates the same cooked chunks and cooked mirror chunks on the target system that are on the source server. The ifxclone utility is not meant to configure all of the possible configuration options, but rather to provide enough configuration to clone the source server.

The number of CPU VPs and buffers on the target server can be configured using the size parameter. Table 132: List of size parameter values on page 419 lists the number of CPU VPs and buffer pools created on the target server for each size option. Additional refinement of the generated configuration should be performed after the target system is created. If the size configuration is omitted, the parameter configured on the source server is used.

Table 132. List of size parameter values

Size	Number of CPU VPs	Number of buffers
tiny	1	50,000
small	2	100,000
medium	4	250,000

Table 132. List of size parameter values (continued)

Size	Number of CPU VPs	Number of buffers
large	8	500,000

You can use the -c option to specify a configuration parameter and its value on the target server. You can also use an existing configuration file. If the target server contains a configuration file that is different than the source server configuration file, the ifxclone utility does not overwrite the file but modifies those parameters that must match the source server during the cloning process.

The --useLocal parameter is required if the target server is located on the same host machine as the source server.

If the --useLocal parameter is specified, the ifxclone utility merges the source server `onconfig` file with the target server `onconfig` file. The configuration parameters listed in Prerequisites for all servers on page 420 are overwritten by the ifxclone utility and the rest of the parameters are not affected.

If the --useLocal parameter is not specified as an input parameter, the ifxclone utility uses the source server's `onconfig` file as the target's `onconfig` file and uses the server name from the input parameters of the ifxclone utility.

If the --useLocal parameter is not specified, the ifxclone utility updates the `sqlhosts` file on the host server with the target server entry and copies both entries to the target's `sqlhosts` file.

The order of precedence of options for the ifxclone parameters is as follows:

- The --configParm parameter takes precedence over the configuration file on the source server.
- The --size parameter takes precedence over merged configuration parameters or the settings in the local configuration file.
- The --configParm parameter takes precedence over the --size parameter.
- Parameters that must be the same on each server take precedence over all other options.

Prerequisites for all servers

Perform the following prerequisites before cloning a server:

- Hardware and software requirements for the servers are generally the same as those for HDR secondary servers (refer to the machine notes for specific supported platforms).
- Both the source and target servers must be part of a trusted network environment. See Network security files on page for information about configuring a trusted environment.
- If the disposition of the target server is specified as ER or RSS then you must provide users with connection permission to the **sysadmin** database on the source server. By default, connection permission to the sysadmin database is limited to user **informix**.
- Only one server clone process can occur at a time. Do not start cloning a second server until the first clone process has completed running.

- The source server must have the ENABLE_SNAPSHOT_COPY configuration parameter set to `1` in the `onconfig` file.
- The target server must not have any old ROOTPATH pages. If the target server has old ROOTPATH pages, create a zero-length `ROOTPATH` file or set the FULL_DISK_INIT configuration parameter to 1 in the target server's `onconfig` file.
- The target server must not have any existing storage space encryption keystore or stash files. If you receive an error message that the command failed because of existing keystore and stash files, follow the instructions in the message and rerun the ifxclone command.

Archive operations, such as ON-Bar command, is not allowed while cloning a server. Perform your data archive activities before starting to clone a server.

The following environment variables must be set on the target server before cloning a server:

- ONEDB_HOME
- ONEDB_SERVER
- ONEDB_ SQLHOSTS
- ONCONFIG

The following configuration parameter values must be identical on both the source and target servers:

- DISK_ENCRYPTION (if the target server is an SD secondary server)
- DRAUTO
- DRINTERVAL
- DRTIMEOUT
- LOGBUFF
- LOGFILES
- LOGSIZE
- LTAPEBLK
- LTAPESIZE
- ROOTNAME
- ROOTSIZE
- PHYSBUFF
- PHYSFILE
- STACKSIZE
- TAPEBLK
- TAPESIZE

If the MIRROR configuration parameter is enabled on the target server, the following configuration parameters also must match between the source and target servers:

- MIRRORPATH
- MIRROROFFSET

The following table shows the valid combinations of the MIRROR configuration parameter on the source and target servers.

Table 133. Valid settings of the MIRROR configuration parameter on source and target servers

MIRROR configuration parameter set on the source server	MIRROR configuration parameter set on the target server	Permitted or not permitted
No	No	Permitted
Yes	Yes	Permitted
Yes	No	Permitted
No	Yes	Not permitted. If this setting is configured, the server issues a warning and disables the MIRROR parameter in the target server onconfig file.

Prerequisites for cloning an RS secondary server

1. Set the following environment variables on the target server:
   - ONEDB_HOME
   - ONEDB_SERVER
   - ONCONFIG
   - ONEDB_ SQLHOSTS
2. On the target server, create all of the chunks and mirror chunks that exist on the source server. If the target server is using mirroring, the mirror chunk paths must match those of the source server and the chunks must exist. You can use the --createchunkfile option (-k) to automatically create cooked chunks on the target server. Follow these steps to create the chunks and (if necessary) mirror chunks for the target server:
   a. On the source server, run the onstat -d command to display a list of chunks and mirror chunks:
      ```
      onstat -d
      ```
   b. On the target server, log in as user informix and use the touch, chown, and chmod commands to create the set of chunks and mirror chunks reported by the onstat -d command. For example, to create a chunk named `/usr/informix/chunks/rootdbs.chunk`, follow these steps:
      ```
      $ su informix
      Password:
      $ touch /usr/informix/chunks/rootdbs.chunk
      $ chown informix:informix /usr/informix/chunks/rootdbs.chunk
      $ chmod 660 /usr/informix/chunks/rootdbs.chunk
      ```
   c. Repeat all of the commands in the previous step for each chunk reported by the onstat -d command.
3. Run the ifxclone utility with the appropriate parameters on the target system.
4. Optionally, create `onconfig` and `sqlhosts` files on the target server.

Example

Example 1, Cloning an RS secondary server using the source server configuration

This example shows how to clone a server by using the `onconfig` and `sqlhosts` configuration files from the source server.

In this example, omitting the -L option causes the ifxclone utility to retrieve the necessary configuration information from the source server. The configuration files are used as a template to create the target server configuration. Having the ifxclone utility create the configuration files for you saves time and reduces the chance of introducing errors into the configuration files.

The -k option creates the necessary cooked chunks on the target server.

For this example, assume that the source server (Amsterdam) has an `sqlhosts` file configured as follows:

```
#Server   Protocol HostName        Service Group
Amsterdam onsoctcp 192.168.0.1 123     -
```

You also need the name, IP address, and port number of the target server. The following values are used for this example:

- Source server name: Amsterdam
- Source IP address: 192.168.0.1
- Source port: 123
- Target server name: Berlin
- Target IP address: 192.168.0.2
- Target port: 456

1. On the target server, create all of the chunks that exist on the source server. You can use the --createchunkfile option (-k) to automatically create cooked chunks on the target server. Log-in as user **informix** and use the commands touch, chown, and chmod to create the chunks.
2. On the target server, run the ifxclone utility:

   ```
   ifxclone -T -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
            -i 192.168.0.2 -p 456 -d RSS -k
   ```

   ```
   ifxclone -T -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
            -i 192.168.0.2 -p 456 -d RSS
   ```

   The ifxclone utility modifies the `sqlhosts` file on the source server and creates a copy of the file on the new target server. The `sqlhosts` file on the target server is the same as the source server:

   ```
   #Server   Protocol HostName        Service Group
   Amsterdam onsoctcp 192.168.0.1 123     -
   Berlin    onsoctcp 192.168.0.2 456
   ```

Example

Example 2, Cloning an RS secondary server by merging the source server configuration

Use the --useLocal option to create a clone of a server on a remote host computer: The --useLocal option is used to merge the source `onconfig` file configuration information with the target `onconfig` file. This option also copies the source `sqlhosts` file to the target server. The following values are used for this example:

- Source server name: Amsterdam
- Source IP address: 192.168.0.1
- Source port: 123
- Target server name: Berlin
- Target IP address: 192.168.0.2
- Target port: 456

1. Create the `onconfig` and `sqlhosts` files and set the environment variables on the target computer.
2. On the target server, create all of the chunks that exist on the source server. You can use the --createchunkfile option (-k) to automatically create cooked chunks on the target server. Log-in as user **informix** and use the commands touch, chown, and chmod to create the chunks.
3. On the target server, run the ifxclone utility:

```
ifxclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
        -i 192.168.0.2 -p 456 -d RSS -k
```

```
ifxclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
        -i 192.168.0.2 -p 456 -d RSS
```

Example

Example 3, Adding an RS secondary server to a cluster

This example shows how to add an RS secondary server to the existing HCL OneDB™ high-availability cluster. The following values are used for this example:

- Source server name: Amsterdam
- Source IP address: 192.168.0.1
- Source port: 123
- Target server name: Berlin
- Target IP address: 192.168.0.2
- Target port: 456

1. Create the `onconfig` and `sqlhosts` files and set the environment variables on the target computer.
2. On the target server, create all of the chunks that exist on the source server. You can use the --createchunkfile option (-k) to automatically create cooked chunks on the target server. Log-in as user **informix** and use the commands touch, chown, and chmod to create the chunks.
3. On the target server, run the ifxclone utility:

```
ifxclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
        -i 192.168.0.2 -p 456 -s medium -d RSS -k
```

```
ifxclone -T -L -S Amsterdam -I 192.168.0.1 -P 123 -t Berlin
        -i 192.168.0.2 -p 456 -s medium -d RSS
```

Prerequisites for cloning an ER server

Complete the following prerequisites before attempting to clone an ER server.

1. The source server (that is, the server that is being cloned) must have ER configured and active.
2. For configuration parameters that specify directory names, the directory names must exist on the target server. For example, if the CDR_LOG_STAGING_DIR configuration parameter is set to a directory name on the source server then the directory must also exist on the target server.
3. If ATS or RIS is enabled on the source server then the appropriate ATS or RIS directories must exist on the target server. See Enabling ATS and RIS File Generation on page and Creating ATS and RIS directories on page . If the directories do not exist then ATS/RIS spooling will fail.
4. If the source server has the CDR_SERIAL configuration parameter set then you must set the value for CDR_SERIAL to a different value on the server to be cloned. The value of CDR_SERIAL must be different on all replication servers. You can specify a unique value for the CDR_SERIAL configuration parameter by using the --configParm (-c) parameter in the ifxclone command line.
5. The clock on the new ER clone must be appropriately synchronized. See Time synchronization on page .
6. The source server (that is, the server being cloned) must not have any stopped or suspended replicates, nor can it have any shadow replicates defined.

Avoid performing ER administrative tasks that change the set of replicates on which the target server participates while the ifxclone utility is running.

Example

Example: Creating a clone of an ER server

Suppose you have five ER servers named S1, S2, S3, S4, and S5 currently configured as root servers in an ER domain. You would like to add a new server, S6, on a new computer named machine6, and you want it to have the same data as server S3.

1. Install and configure HCL OneDB™ database software on machine6. You can use the deployment utility to deploy a pre-configured database server instance.
2. Copy the `sqlhosts` file from server S3 to server S6 and modify it to add entries for the new server. For example, assuming the ER group name for the new server is g_S6 and the ID is 60, the `sqlhosts` file lines would look like the following.

   ```
   g_S6    group     -           -          i=60
   S6      onsoctcp  machine6    service6   g=g_S6
   ```

3. Add the two lines from the previous step in the `sqlhosts` files on all of the other five servers (S1 through S5).
4. Copy the `onconfig` file from server S3 to server S6 and change the DBSERVERNAME configuration parameter to S6. Do not modify any storage or chunk parameters except for path information.
5. On server S6 (machine6) provision chunk paths and other storage to the same sizes as server S3. Ensure that S6 has adequate memory and disk space resources. You can use the --createchunkfile option (-k) to automatically create cooked chunks on the target server.
6. Run the following command as user **informix**:

   ```
   ifxclone -L -S S3 -I machine3 -P service3 -t S6 -i machine6 -p service6 -d ER -k
   ```

   ```
   ifxclone -L -S S3 -I machine3 -P service3 -t S6 -i machine6 -p service6 -d ER
   ```

   When prompted, enter the user name **informix** and then enter the password for user **informix**.

7. Monitor the server logs of servers S6 and S3. When the cloning process is complete you can check the status of servers by running the following command on servers S3 and S6:

```
cdr list server
```

You should see the new ER server g_S6 connected to all of the other five servers. In addition, ER node g_S6 will now participate in all replicates in which ER node g_S3 participates.

The onspaces utility

Use the onspaces utility to manage the storage spaces in your database.

onspaces syntax

Run onspaces utility commands to manage your storage spaces.

```
>>-onspaces--+-------------------------+--+- -a-------+--------><
             |                     (1) |  +- -c -b----+
             '-|  -FILE option    |-----'  +- -c -d----+
                                          +- -c -P----+
                                          +- -c -S----+
                                          +- -c -x----+
                                          +- -ch------+
                                          +- -cl------+
                                          +- -d-------+
                                          +- -f-------+
                                          +- -m-------+
                                          +- -r-------+
                                          +- -ren-----+
                                          +- -s-------+
                                          +- -V-------+
                                          '- -version-'
```

Element	Purpose	Key Considerations
-V	Shows the software version number and the serial number	See Obtaining utility version information on page 312
-version	Shows the build version, host, OS, build number, date, and the GLS version	See Obtaining utility version information on page 312

onspaces -a: Add a chunk to a dbspace or blobspace

```
>>-onspaces -a--+-dbspace---+--+- -p--pathname--------------+--->
                '-blobspace-'  |                        (1) |
                               '- -p--\--\--.--\--drive-----'

>-- -o--offset-- -s--size--------------------------------------->

>--+- -m--pathname offset-------------+-----------------------><
   |                              (1) |
   '- -m--\--\--.--\--drive offset-----'
```

Use onspaces -a to add a chunk to a dbspace or blobspace.

Element	Purpose	Key considerations
-a	Indicates that a chunk is to be added	You can have up to 32766 chunks in an instance. You can put all those chunks in one storage space, or spread them among multiple storage spaces.
drive	Specifies the Windows™ drive to allocate as unbuffered disk space The format can be either ***\\.\<drive>***, where ***drive*** is the drive letter assigned to a disk partition, or ***\\.\PhysicalDrive<number>***, where ***PhysicalDrive*** is a constant value and ***number*** is the physical drive number.	For more information, see Allocating raw disk space on Windows on page . Example: `\\.\F:` For path name syntax, see your operating-system documentation.
-m ***pathname offset***	Specifies an optional path name and offset to the chunk that mirrors the new chunk Also see the entries for *pathname* and *offset* in this table.	For more information, see Adding a chunk to a dbspace or blobspace on page .
-o ***offset***	After the **-a** option, *offset* indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace or dbspace	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes. For more information, see Allocating raw disk space on UNIX on page .
-p ***pathname***	Indicates the disk partition or unbuffered device of the initial chunk of the blobspace or dbspace that you are adding The chunk must be an existing unbuffered device or buffered file.	The chunk path name can be up to 256 bytes. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. UNIX™ example (unbuffered device): `/dev/rdsk/c0t3d0s4` UNIX™ example (buffered device): `/ix/ids9.2/db1chunk` Windows™ example: `c:\Ifmxdata\ol_icecream\mychunk1.dat` For path name syntax, see your operating-system documentation.
-s ***size***	Indicates, in kilobytes, the size of the new blobspace or dbspace chunk	Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus

Element	Purpose	Key considerations
		the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes.
blobspace	Names the blobspace to which you are adding a chunk	See Adding a chunk to a dbspace or blobspace on page . Syntax must conform to the Identifier on page .
dbspace	Names the dbspace to which you are adding a chunk	See Adding a chunk to a dbspace or blobspace on page . Syntax must conform to the Identifier on page .

This command has an equivalent SQL administration API function.

onspaces -a: Add a chunk to an sbspace

```
>>-onspaces -a--sbspace-- -p--pathname-- -o--offset------------->

>-- -s--size--+----------------------+--+-------------+-------->
              '- -m--pathname offset-'  '- -Ms--mdsize-'

>--+---------------+--+-----+-------------------------------><
   '- -Mo--mdoffset-'  '- -U-'
```

Use **onspaces -a** to add a chunk to an sbspace.

Element	Purpose	Key considerations
-a	Indicates that a chunk is to be added	You can have up to 32766 chunks in an instance. You can put all those chunks in one storage space, or spread them among multiple storage spaces.
-m *pathname offset*	Specifies an optional path name and offset to the chunk that mirrors the new chunk Also see the entries for ***pathname*** and ***offset*** in this table.	For background information, see adding a chunk to an sbspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-Mo *mdoffset*	Indicates, in kilobytes, the offset into the disk partition or into the device where metadata should be stored	Value can be an integer between 0 and the chunk size. You cannot specify an offset that causes the end of the metadata space to be past the end of the chunk. For background information, see sizing sbspace metadata, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-Ms *mdsize*	Specifies the size, in kilobytes, of the metadata area allocated in the initial	Value can be an integer between 0 and the chunk size.

Element	Purpose	Key considerations
	chunk. The remainder is user-data space	For background information, see sizing sbspace metadata, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-o *offset*	After the **-a** option, *offset* indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the new blobspace or dbspace.	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 terabytes, depending on the platform. For more information, see allocating raw disk space on UNIX™, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-p *pathname*	Indicates the disk partition or unbuffered device of the initial chunk of the sbspace that you are creating The chunk must be an existing unbuffered device or buffered file.	The chunk path name can be up to 256 bytes. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. For path name syntax, see your operating-system documentation.
-U	Specifies that the entire chunk should be used to store user data	The **-M** and **-U** options are mutually exclusive. For background information, see adding a chunk to an sbspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-s *size*	Indicates, in kilobytes, the size of the new sbspace chunk	Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes
sbspace	Names the sbspace to which you are adding a chunk	See adding a chunk to an sbspace in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*. Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*.

This command has an equivalent SQL administration API function.

onspaces -c -b: Create a blobspace

Syntax:

(explicit id unique_492_Connect_42_blob01) unique_492_Connect_42_blob01 `onspaces -c` { `-b`*blobspace* `-g` *pageunit* } { `-p`*pathname* | `-p\\.\` *drive* } `-o`*offset* `-s`*size* [{ `-m`*pathname offset* | `-m\\.\`*drive offset* }] [`-u`]

Use onspaces -c -b to create a blobspace.

Element	Purpose	Key considerations
-b *blobspace*	Names the blobspace to be created	The blobspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the $ character. For more information, see creating a blobspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*. The syntax must conform to the Identifier segment. For more information, see the *HCL OneDB™ Guide to SQL: Syntax*.
-c	Creates a dbspace, blobspace, sbspace, or extspace You can create up to 2047 storage spaces of any type.	After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. For more information, see creating a dbspace, blobspace, or extspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.

drive	Specifies the Windows™ drive to allocate as unbuffered disk space
	
	The format can be either \\.\<*drive*>, where *drive* is the drive letter assigned to a disk partition, or \\.*PhysicalDrive<number>*, where *PhysicalDrive* is a constant value and *number* is the physical drive number.
		For information on allocating unbuffered disk space, see allocating unbuffered disk space on Windows™ in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*. Examples:
		`\\.\F:`
		`\\.\PhysicalDrive2`
		
		For path name syntax, see your operating-system documentation.

-g *pageunit*	Specifies the blobspace blobpage size in terms of *page_unit*, the	Unsigned integer. Value must be greater than 0.

Element	Purpose	Key considerations
	number of the base page size of the instance (either 2K or 4K)	The maximum number of pages that a blobspace can contain is 2147483647. Therefore, the size of the blobspace is limited to the blobpage size x 2147483647. This includes blobpages in all chunks that make up the blobspace. For more information, see blobpage size considerations, in the chapter on I/O Activity in the *HCL OneDB™ Performance Guide*.
-m *pathname offset*	Specifies an optional path name and offset to the chunk that mirrors the initial chunk of the new blobspace or dbspace Also see the entries for -p *pathname* and -o *offset* in this table.	For more information, see creating a dbspace or a blobspace in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-o *offset*	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace, dbspace, or sbspace	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 terabytes, depending on the platform. For more information, see allocating raw disk space, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-p *pathname*	Indicates the disk partition or device of the initial chunk of the blobspace or dbspace that you are creating	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. UNIX™ example (unbuffered device): `/dev/rdsk/c0t3d0s4` UNIX™ example (buffered device): `/ix/ids9.2/db1chunk` Windows™ example:`c:\Ifmxdata\ol_icecream\mychunk1.dat` For path name syntax, see your operating-system documentation.
-s *size*	Indicates, in kilobytes, the size of the initial chunk of the new blobspace or dbspace	Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting

Element	Purpose	Key considerations
		offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.
-u	Specifies to create an unencrypted space	Use this option to create an unencrypted storage space when encryption is enabled by the DISK_ENCRYPTION configuration parameter.

This command has an equivalent SQL administration API function.

onspaces -c -d: Create a dbspace

Use the onspaces -c -d command to create a dbspace or a temporary dbspace.

Syntax

(explicit id unique_493_Connect_42_note020) unique_493_Connect_42_note020 **onspaces -c** { **-d** *dbspace* } { **-p** *pathname* | **-p** \ \ . \ *drive* } **-o** *offset* **-s** *size* [{ [**-ef** *extentsize* **-en** *extentsize*] | **-t** }] [{ **-m** *pathname offset* | **-m** \ \ . \ *drive offset* }] [**-k** *pagesize*] [**-u**]

Element	Purpose	Key considerations
-c	Creates a dbspace You can create up to 2047 storage spaces of any type.	After you create a storage space, you must back up both this storage space and the root dbspace. If you create a storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. For more information, see Manage dbspaces on page .

drive	Specifies the Windows™ drive to allocate as unbuffered disk space
	
	The format can be either `\\.\drive`, where *drive* is the drive letter that is assigned to a disk partition, or `\\.\PhysicalDrivenumber`, where *PhysicalDrive* is a constant
		For information on allocating unbuffered disk space, see Allocating raw disk space on Windows on page .
		
		Examples:
		
		`\\.\F:`
		`\\.\PhysicalDrive2`
		
		For path name syntax, see your operating-system documentation.

Element	Purpose	Key considerations
	value and *number* is the physical drive number.	
-d *dbspace*	Names the dbspace to be created	The dbspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the $ character. For more information, see Manage dbspaces on page . The syntax must conform to the Identifier segment. For more information, see Identifier on page .
-ef *extentsize*	Indicates, in KB, the size of the first extent for the tblspace **tblspace**	The minimum, and default, size of the first extent for the tblspace **tblspace** of a non-root dbspace is equivalent to 50 dbspace pages, which are specified in KB. For example: 100 KB for a 2 KB page size dbspace, 200 KB for a 4 KB page size dbspace, 400 KB for an 8 KB page size dbspace. The maximum size of a tblspace **tblspace** extent is 1048575 pages minus the space that is needed for any system objects. On a 2 KB page size system, the maximum size is approximately 2 GB. For more information, see Specifying the first and next extent sizes for the tblspace tblspace on page .
-en *extentsize*	Indicates, in KB, the size of the next extents in the tblspace **tblspace**	The minimum size of the next extents for the tblspace **tblspace** of a non-root dbspace is equivalent to 4 dbspace pages, which are specified in KB. For example: 8 KB for a 2 KB page size dbspace, 16 KB for a 4 KB page size dbspace, 32 KB for an 8 KB page size dbspace. The default size for a next extent is 50 dbspace pages. The maximum size of a tblspace **tblspace** extent is 1048572 pages. On a 2 KB page size system, the maximum size is approximately 2 GB. If there is not enough space for a next extent in the primary chunk, the extent is allocated from another chunk. If the specified space is not available, the closest available space is allocated.

Element	Purpose	Key considerations
		For more information, see Specifying the first and next extent sizes for the tblspace tblspace on page .
-k *pagesize*	Indicates in KB, the non-default page size for the new dbspace. For systems with sufficient storage, performance advantages of a larger page size can include the following: • Reduced depth of B-tree indexes, even for smaller index keys • You can group on the same page long rows that currently span multiple pages of the default page size • Checkpoint time is typically reduced with larger pages • You can define a different page size for temporary tables so that they have a separate buffer pool.	The page size must be between 2 KB and 16 KB and must be a multiple of the default page size. For example, if the default page size is 2 KB, then *pagesize* can be 2, 4, 6, 8, 10, 12, 14, or 16. If the default page size is 4 KB (Windows™), then *pagesize* can be 4, 8, 12, or 16. For more information, see Creating a dbspace with a non-default page size on page .
-m *pathname offset*	Specifies an optional path name and offset to the chunk that mirrors the initial chunk of the new dbspace Also see the entries for -p *pathname* and -o *offset* in this table.	For more information, see Manage dbspaces on page .
-o *offset*	Indicates, in KB, the offset into the disk partition or into the device to reach the initial chunk of the new dbspace	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The offset must be a multiple of the page size. The maximum offset is 2 or 4 TB, depending on the platform. For more information, see Allocating raw disk space on Windows on page .

Element	Purpose	Key considerations
-p *pathname*	Indicates the disk partition or device of the initial chunk of the dbspace that you are creating	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. UNIX™ example (unbuffered device): `/dev/rdsk/c0t3d0s4` UNIX™ example (buffered device): `/ix/ids9.2/db1chunk` Windows™ example:`c:\Ifmxdata\ol_icecream\mychunk1.dat` For path name syntax, see your operating-system documentation.
-s *size*	Indicates, in KB, the size of the initial chunk of the new dbspace	Unsigned integer. The size must be equal to or greater than 1000 KB and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 TB, depending on the platform.
-t	Creates a temporary dbspace for storage of temporary tables	You cannot mirror a temporary dbspace. You cannot specify the first and next extent sizes for the tblspace **tblspace** of a temporary dbspace. For more information, see Temporary dbspaces on page .
-u	Specifies to create an unencrypted space	Use this option to create an unencrypted storage space when encryption is enabled by the DISK_ENCRYPTION configuration parameter.

The maximum size of a dbspace is equal to the maximum number of chunks multiplied by the maximum size of a chunk. (The maximum number of chunks is 32766 per instance. The maximum size of a chunk is equal to 2147483647 pages multiplied by the page size.)

This command has an equivalent SQL administration API function.

You cannot change the page size of a dbspace after you create it.

You cannot store logical or physical logs in a dbspace that is not the default platform page size.

If a dbspace is created when a buffer pool with that page size does not exist, HCL OneDB™ creates a buffer pool using the values of the fields of the default line of the BUFFERPOOL parameter.

Temporary dbspaces

When you create a temporary dbspace with onspaces, the database server uses the newly created temporary dbspace, after you add the name of the new temporary dbspace to your list of temporary dbspaces in the DBSPACETEMP configuration parameter, the DBSPACETEMP environment variable, or both and restart the server.

You cannot specify the first and next extent of a temporary dbspace. The extent size for temporary dbspaces is 100 KB for a 2 KB page system or 200 KB for a 4 KB page system.

You can specify the first and next space of the tblspace **tblspace** in the root dbspace if you do not want the database server to automatically manage the size. To specify the first and next extent sizes of a root tblspace **tblspace**, use the TBLTBLFIRST and TBLTBLNEXT configuration parameters before you create the root dbspace the first time that you start the database server.

onspaces -c -P: Create a plogspace

Use the onspaces -c -P command to create a plogspace in which to store the physical log.

Syntax

(explicit id unique_494_Connect_42_1182note020) unique_494_Connect_42_1182note020 **onspaces -c** { **-P** *plogspace* } { **-p** *pathname* | **-p** \ \ . \ *drive* } **-o** *offset* **-s** *size* [{ **-m** *pathname offset* | **-m** \ \ . \ *drive offset* }] [**-u**]

Element	Purpose	Key considerations
-c	Creates a plogspace.	An instance can have only one plogspace. If a plogspace exists, creating a new one moves the physical log to the new space and drops the old plogspace.
-m *pathname offset*	Specifies an optional path name and offset to the chunk that mirrors the chunk of the new plogspace. See -p *pathname* and -o *offset* in this table.	If you mirror the plogspace, the plogspace chunk cannot be extendable.

-m `\\.\`*drive*	Specifies the Windows™ drive for the chunk that mirrors the chunk of the new plogspace.
	
	The *drive* is the drive letter that is assigned to a disk partition or a constant value and the physical drive number.
		Examples:
		`\\.\F:`
		`\\.\PhysicalDrive2`
		
		For drive name syntax, see your operating-system documentation.

Element	Purpose	Key considerations
-o *offset*	Indicates, in KB, the offset into the disk partition or into the device to reach the chunk of the new plogspace.	Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The offset must be a multiple of the page size. The maximum offset is 2 or 4 TB, depending on the platform.
-P *plogspace*	Names the plogspace to be created.	The plogspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the $ character. The syntax must conform to the Identifier segment. For more information, see Identifier on page .

-p *pathname*	Indicates the disk partition or device of the chunk of the plogspace that you are creating.	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server.
		
		UNIX™ example (unbuffered device):
		
		`/dev/rdsk/c0t3d0s4`
		
		UNIX™ example (buffered device):
		
		`/ix/ifmx/db1chunk`
		
		Windows™ example:
		
		`c:\Ifmxdata\ol_icecream\mychunk1.dat`

-p `\\.\`*drive*	Specifies the Windows™ drive to allocate as unbuffered disk space for the plogspace.
	
	The *drive* is the drive letter that is assigned to a disk partition or a constant value and the physical drive number.
		Examples:
		
		`\\.\F:`
		`\\.\PhysicalDrive2`
		
		For drive name syntax, see your operating-system documentation.

-s *size*	Indicates, in KB, the size of the chunk of the new plogspace.	Unsigned integer. The size must be equal to or greater than 1000 KB and a multiple of the page size. The starting offset

Element	Purpose	Key considerations
		plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 TB, depending on the platform.
-u	Specifies to create an unencrypted space	Use this option to create an unencrypted storage space when encryption is enabled by the DISK_ENCRYPTION configuration parameter.

The physical log must be stored on a single chunk. By default the chunk for the plogspace is extendable and the database server expands the plogspace as needed to improve performance.

Example

Examples

The following example creates a plogspace that is called **plogdbs** that has a size of 40000 KB and an offset of 0:

```
onspaces -c -P plogdbs -p /dev/chk1 -o 0 -s 40000
```

The following example creates a mirrored plogspace that is called **pdbs1** that has a size of 60000 KB and an offset of 500 KB:

```
onspaces -c -P pdbs1 -p /dev/pchk1 -o 500 -s 60000 -m /dev/mchk1 0
```

onspaces -c -S: Create an sbspace

Use the onspaces -c -S option to create a sbspace or a temporary sbspace.

Syntax:

onspaces -c -S*sbspace* [-t] -p*pathname* -o*offset* -s*size* [-m*pathname offset*] [-Ms*mdsize*] [-Mo*mdoffset*] [-Df*default list*] [-u]

Element	Purpose	Key Considerations
-S*sbspace*	Names the sbspace to be created	The sbspace name must be unique and must not exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the $ character. Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*.
-c	Creates an sbspace	None.

Element	Purpose	Key Considerations
	You can create up to 32767 storage spaces of any type.	
-Df *default list*	Lists default specifications for smart large objects stored in the sbspace	Restrictions: Tags are separated by commas. If a tag is not present, system defaults take precedence. The list must be enclosed in double quotation marks (") on the command line. References: For a list of tags and their parameters, see Table 134: -Df Default Specifications on page 441.
-m *pathname offset*	Specifies an optional pathname and offset to the chunk that mirrors the initial chunk of the new sbspace Also see the entries for -p *pathname* and -o *offset* in this table.	For more information, see sbspaces in the chapter on data storage, and creating an sbspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-Mo *mdoffset*	Indicates, in kilobytes, the offset into the disk partition or into the device where metadata will be stored.	Restrictions: Value can be an integer between 0 and the chunk size. You cannot specify an offset that causes the end of the metadata space to be past the end of the chunk. References: For more information, see sizing sbspace metadata, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-Ms *mdsize*	Specifies the size, in kilobytes, of the metadata area allocated in the initial chunk The remainder is user-data space.	Restrictions: Value can be an integer between 0 and the chunk size.
-o *offset*	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the sbspace	Restrictions: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 4 terabytes for systems with a two-kilobyte page size and 8 terabytes for systems with a four-kilobyte page size. References: For more information, see allocating raw disk space on UNIX™, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.

Element	Purpose	Key Considerations
-p *pathname*	Indicates the disk partition or unbuffered device of the initial chunk of the sbspace	The chunk must be an existing unbuffered device or buffered file. When you specify a pathname, you can use either a full pathname or a relative pathname. However, if you use a relative pathname, it must be relative to the directory that was the current directory when you initialized the database server. References: For pathname syntax, see your operating-system documentation.
-s *size*	Indicates, in kilobytes, the size of the initial chunk of the new sbspace	Restrictions: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.
-t	Creates a temporary sbspace for storage of temporary smart large objects. You can specify the size and offset of the metadata area	Restrictions: You cannot mirror a temporary sbspace. You can specify any -Df option, except the LOGGING=ON option, which has no effect. References: For more information, see Creating a Temporary Sbspace with the -t Option on page 440.
-u	Specifies to create an unencrypted space	Use this option to create an unencrypted storage space when encryption is enabled by the DISK_ENCRYPTION configuration parameter.

Creating a Temporary Sbspace with the -t Option

This example creates a temporary sbspace of 1000 kilobytes:

```
onspaces -c -S tempsbsp -t -p ./tempsbsp -o 0 -s 1000
```

You can optionally specify the name of the temporary sbspace in the SBSPACETEMP configuration parameter. Restart the database server so that it can use the temporary sbspace.

Creating an Sbspace with the -Df option

When you create an sbspace with the optional **-Df** option, you can specify several default specifications that affect the behavior of the smart large objects stored in the sbspace. The default specifications must be expressed as a list separated by commas. The list need not contain all of the tags. The list of tags must be enclosed in double quotation marks. The table in Table 134: -Df Default Specifications on page 441 describes the tags and their default values.

The four levels of inheritance for sbspace characteristics are system, sbspace, column, and smart large objects. For more information, see smart large objects in the chapter on where data is stored in the *HCL OneDB™ Administrator's Guide*.

Table 134. -Df Default Specifications

Tag	Values	Default	Description
ACCESSTIME	ON or OFF	OFF	When set to ON, the database server tracks the time of access to all smart large objects stored in the sbspace. For information about altering storage characteristics of smart large objects, see the *HCL OneDB™ DataBlade® API Programmer's Guide*.
AVG_LO_SIZE	Windows™: 4 to 2**31 UNIX™: 2 to 2**31	8	Specifies the average size, in kilobytes, of the smart large object stored in the sbspace The database server uses this value to calculate the size of the metadata area. Do not specify AVG_LO_SIZE and **-Ms** together. You can specify AVG_LO_SIZE and the metadata offset (**-Mo**) together. If the size of the smart large object exceeds 2**31, specify 2**31. If the size of the smart large object is less than 2 on UNIX™ or less than 4 in Windows™, specify 2 or 4. Error 131 is returned if you run out of space in the metadata and reserved areas in the sbspace. To allocate additional chunks to the sbspace that consist of metadata area only, use the **-Ms** option instead. For more information, see creating smart large objects, in the chapter on managing data on disk in the *HCL OneDB™ Administrator's Guide*.
BUFFERING	ON or OFF	ON	Specifies the buffering mode of smart large objects stored in the sbspace If set to ON, the database server uses the buffer pool in the resident portion of shared memory for smart-large-object I/O operations. If set to OFF, the database server uses light I/O buffers in the virtual portion of shared memory (lightweight I/O operations). BUFFERING = OFF is incompatible with LOCK_MODE = RANGE and creates a conflict

Table 134. -Df Default Specifications (continued)

Tag	Values	Default	Description
			For more information, see lightweight I/O, in the chapter on configuration effects on memory in the *HCL OneDB™ Performance Guide*.
LOCK_MODE	RANGE or BLOB	BLOB	Specifies the locking mode of smart large objects stored in the sbspace If set to RANGE, only a range of bytes in the smart large object is locked. If set to BLOB, the entire smart large object is locked. LOCK_MODE = RANGE is incompatible with BUFFERING = OFF and creates a conflict. For more information, see smart large objects, in the chapter on locking in the *HCL OneDB™ Performance Guide*.
LOGGING	ON or OFF	OFF	Specifies the logging status of smart large objects stored in the sbspace If set to ON, the database server logs changes to the user data area of the sbspace. When you turn on logging for an sbspace, take a level-0 backup of the sbspace. When you turn off logging, the following message displays: You are turning off smart large object logging. For more information, see smart large objects, in the chapters on data storage and logging in the *HCL OneDB™ Administrator's Guide*. For information about **onspaces -ch** messages, see Messages in the database server log.
EXTENT_SIZE	4 to 2**31	None	Specifies the size, in kilobytes, of the first allocation of disk space for smart large objects stored in the sbspace when you create the table Let the system select the EXTENT_SIZE value. To reduce the number of extents in a smart large object, use **mi_lo_specset_estbytes** (DataBlade® API) or **ifx_lo_specset_estbytes** () to hint to the system the total size of the smart large object. The system attempts to allocate a single extent for the smart large object.

Table 134. -Df Default Specifications (continued)

Tag	Values	Default	Description
			For more information, see smart large objects, in the chapter on where data is stored in the *HCL OneDB™ Administrator's Guide*. For information about altering storage characteristics of smart large objects, see the *HCL OneDB™ DataBlade® API Programmer's Guide* or the *HCL OneDB™ ESQL/C Programmer's Manual*.
MIN_EXT_SIZE	2 to 2**31	Windows™: 4UNIX: 2	Specifies the minimum amount of space, in kilobytes, to allocate for each smart large object The following message displays: Changing the sbspace minimum extent size: old value *value1* new value *value2*. For information about tuning this value, see smart large objects, in the chapter on configuration effects on I/O utilization in the *HCL OneDB™ Performance Guide*. For information about **onspaces -ch** messages, see Messages in the database server log.
NEXT_SIZE	4 to 2**31	None	Specifies the extent size, in kilobytes, of the next allocation of disk space for smart large objects when the initial extent in the sbspace becomes full. Let the system select the NEXT_SIZE value. To reduce the number of extents in a smart large object, use **mi_lo_specset_estbytes** or **ifx_lo_specset_estbytes** to hint to the system the total size of the smart large object. The system attempts to allocate a single extent for the smart large object. For more information, see smart large objects, in the chapter on where data is stored in the *HCL OneDB™ Administrator's Guide*. For information about obtaining the size of smart large objects, see the *HCL OneDB™ DataBlade® API Programmer's Guide* or the *HCL OneDB™ ESQL/C Programmer's Manual*.

This example creates a 20-megabyte mirrored sbspace, **eg_sbsp,** with the following specifications:

- An offset of 500 kilobytes for the primary and mirror chunks
- An offset of 200 kilobytes for the metadata area
- An average expected smart-large-object size of 32 kilobytes
- Log changes to the smart large objects in the user-data area of the sbspace

UNIX Only:

```
% onspaces -c -S eg_sbsp -p /dev/raw_dev1 -o 500 -s 20000
   -m /dev/raw_dev2 500 -Mo 200 -Df "AVG_LO_SIZE=32,LOGGING=ON"
```

Changing the -Df Settings

About this task

As the database server administrator, you can override or change the **-Df** default settings in one of the following ways:

- To change the default settings for an sbspace, use the **onspaces -ch** option. For more information, refer to onspaces -ch: Change sbspace default specifications on page 446.
- To override the following **-Df** default settings for a specific table, use the SQL statements CREATE TABLE or ALTER TABLE:
    - LOGGING
    - ACCESSTIME
    - EXTENT_SIZE
    - NEXT_SIZE

  For more information on the ALTER TABLE and CREATE TABLE statements, see the *HCL OneDB™ Guide to SQL: Syntax*.

The programmer can override these **-Df** default settings with DataBlade® API and functions. For information about altering storage characteristics of smart large objects, see the *HCL OneDB™ DataBlade® API Programmer's Guide* and the *HCL OneDB™ ESQL/C Programmer's Manual*.

Using the onspaces -g option

The onspaces -g option is not used for sbspaces. The database server uses a different method to determine the number of pages to transfer in an I/O operation for sbspaces than for blobspaces. The database server can automatically determine the block size to transfer in an I/O operation for smart large objects. For more information, see sbspace extent sizes in the chapter on I/O activity in your *HCL OneDB™ Performance Guide*.

This command has an equivalent SQL administration API function.

onspaces -c -x: Create an extspace

Use the onspaces -c -x option to create an extspace.

Syntax:

`onspaces -c -x`*extspace* `-l`*location* `-o`*offset* `-s`*size*

Element	Purpose	Key Considerations
-c	Creates a dbspace, blobspace, sbspace, or extspace	After you create a storage space, you must back up both this storage space and the root dbspace. If you create a

Element	Purpose	Key Considerations
	You can create up to 2047 storage spaces of any type.	storage space with the same name as a deleted storage space, perform another level-0 backup to ensure that future restores do not confuse the new storage space with the old one. For more information, see creating a dbspace, blobspace, or extspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-l *location*	Specifies the location of the extspace The access method determines the format of this string.	**Restrictions**: String. Value must not be longer than 255 bytes. For more information, see creating an extspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-o *offset*	Indicates, in kilobytes, the offset into the disk partition or into the device to reach the initial chunk of the new blobspace, dbspace, or sbspace	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 2 or 4 terabytes, depending on the platform. For more information, see allocating raw disk space, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-s*size*	Indicates, in kilobytes, the size of the initial chunk of the new blobspace or dbspace	**Restrictions**: Unsigned integer. The size must be equal to or greater than 1000 kilobytes and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum chunk size is 2 or 4 terabytes, depending on the platform.
-x *extspace*	Names the extspace to be created	**Restrictions**: Extspace names can be up to 128 bytes. They must be unique, begin with a letter or underscore, and contain only letters, digits, underscores, or **$** characters. For more information, see extspaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.

onspaces -ch: Change sbspace default specifications

Use the **onspaces -ch** option to change the default specifications of a sbspace.

Syntax:

`onspaces -ch` *sbspace* `-Df`*default list*

Element	Purpose	Key Considerations
-ch	Indicates that one or more sbspace default specifications are to be changed	None.
sbspace	Names the sbspace for which to change the default specifications	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For background information, see changing default specifications of an sbspace with **onspaces** in the *HCL OneDB™ Performance Guide*.
-Df *default list*	Lists new default specifications for smart large objects stored in the sbspace	Tags are separated by commas. If a tag is not present, system defaults take precedence. The list must be enclosed in double quotation marks ("") on the command line. For a list of tags and their parameters, see Table 134: -Df Default Specifications on page 441.

You can change any of the **-Df** tags with the **onspaces -ch** option. The database server applies the change to each smart large object that was created prior to changing the default specification.

For example, to turn off logging for the sbspace that you created in Creating an Sbspace with the -Df option on page 440, use the following command:

```
onspaces -ch eg_sbsp -Df "LOGGING=OFF"
```

Note: After you turn on logging for an sbspace, take a level-0 backup of the sbspace to create a point from which to recover.

onspaces -cl: Clean up stray smart large objects in sbspaces

Use the onspaces -cl option to clean up stray smart large objects in sbspaces.

Syntax:

`onspaces -cl` *sbspace*

Element	Purpose	Key Considerations
-cl	Cleans up stray smart large objects in an sbspace	To find any stray smart large objects, use the oncheck -pS command when no users are connected to the database server. The smart large objects with a reference count of 0 are stray objects.
sbspace	Names the sbspace to be cleaned up	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*.

During normal operation, no unreferenced (stray) smart large objects should exist. When you delete a smart large object, the space is released. If the database server fails or runs out of system memory while you are deleting a smart large object, the smart large object might remain as a stray object.

The following is an example of the onspaces -cl command:

```
onspaces -cl myspace
```

The best way to find the reference count for a smart large object is to call the mi_lo_stat or ifx_lo_stat functions from a C program. Although the mi_lo_increfcount and mi_lo_decrefcount functions return the reference count, they increment or decrement the reference count. For more information on these functions, see the *HCL OneDB™ DataBlade® API Function Reference*.

This command has an equivalent SQL administration API function.

onspaces -d: Drop a chunk in a dbspace, blobspace, or sbspace

Use the onspaces -d option to drop a chunk in a dbspace, blobspace, or sbspace.

Syntax:

onspaces -d { *dbspace* | *blobspace* | **[-f]** *sbspace* } **-p***pathname* **-o***offset* [**-y**]

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-d	Drops a chunk	You can drop a chunk from a dbspace, temporary dbspace, or sbspace when the database server is online or quiescent. For more information, see the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*. You can drop a chunk from a blobspace only when the database server is in quiescent mode.
-f	Drops an sbspace chunk that contains user data but no metadata If the chunk contains metadata for	Use the **-f** option with sbspaces only. If you omit the **-f** option, you cannot drop an sbspace that contains data.

Element	Purpose	Key considerations
	the sbspace, you must drop the entire sbspace.	For more information, see dropping a chunk from an sbspace with onspaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-o *offset*	Indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk of the dbspace, blobspace, or sbspace that you are dropping	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes. For more information, see allocating raw disk space on UNIX™, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-p *pathname*	Indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you are dropping	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. For path name syntax, see your operating-system documentation.
-y	Causes the database server to automatically respond yes to all prompts	None.
blobspace	Names the blobspace from which the chunk is dropped	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see dropping a chunk from a blobspace, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
dbspace	Names the dbspace from which the chunk is dropped	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see dropping a chunk from a dbspace with onspaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
sbspace	Names the sbspace from which the chunk is dropped	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For background information, see dropping a chunk from a dbspace with onspaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.

Important: You must specify a path name to indicate to the database server that you are dropping a chunk.

onspaces -d: Drop a space

Use the onspaces -d option to drop a dbspace, blobspace, plogspace, sbspace, or extspace.Use the onspaces -d option to drop a dbspace, blobspace, sbspace, or extspace.

Syntax:

`onspaces -d` { *dbspace* | *blobspace* | *plogspace* | `[ -f]` *sbspace* | *extspace* } [`-y`]

Element	Purpose	Key considerations
-d	Indicates that a storage space is to be dropped	You can drop a dbspace, blobspace, plogspace, sbspace, or extspace while the database server is online or in quiescent mode. You can drop a dbspace, blobspace, sbspace, or extspace while the database server is online or in quiescent mode. After you drop a storage space, you must back it up to ensure that the **sysutils** database and the reserved pages are up-to-date. Run oncheck -pe to verify that no table is storing data in the dbspace, blobspace, or sbspace.
-y	Causes the database server to automatically respond yes to all prompts	None.
-f	Drops an sbspace that contains user data and metadata	You must use the -f (force) option to drop an sbspace that contains data. **Restriction:** Use the -f option with sbspaces only. **Warning:** If you use the -f option, the tables in the database server might have dead pointers to the smart large objects that were deleted with this option.
blobspace	Names the blobspace to be dropped	Before you drop a blobspace, drop all tables that include a TEXT or BYTE column that references the blobspace.
dbspace	Names the dbspace to be dropped	Before you drop a dbspace, drop all databases and tables that you previously created in the dbspace.
extspace	Names the extspace to be dropped	You cannot drop an extspace if it is associated with an existing table or index.

Element	Purpose	Key considerations
plogspace	Names the plogspace to be dropped	The plogspace must be empty to be dropped.
sbspace	Names the sbspace to be dropped	Before you drop an sbspace, drop all tables that include a BLOB or CLOB column that references the sbspace.

Important: Do not specify a path name when you drop these storage spaces.

This command has an equivalent SQL administration API function.

onspaces -f: Specify DATASKIP parameter

Use the onspaces -f option to specify the value of the DATASKIP configuration parameter on a dbspace level or across all dbspaces.

Syntax:

`onspaces -f` { `OFF` | `ON` } [*dbspace-list*] [`-y`]

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-f	Indicates to the database server that you want to change the DATASKIP default for specified dbspaces or all dbspaces	All changes in the DATASKIP status are recorded in the message log.
-y	Causes the database server to automatically respond yes to all prompts	None.
dbspace-list	Specifies the name of one or more dbspaces for which DATASKIP will be turned ON or OFF	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see DATASKIP Configuration Parameter on page 66 and the *HCL OneDB™ Performance Guide*.
OFF	Turns off DATASKIP	If you use OFF without *dbspace-list*, DATASKIP is turned off for all fragments. If you use OFF with *dbspace-list*, only the specified fragments are set with DATASKIP off.
ON	Turns on DATASKIP	If you use ON without *dbspace-list*, DATASKIP is turned on for all fragments. If you use ON with dbspace-list, only the specified fragments are set with DATASKIP on.

onspaces -m: Start mirroring

Use the onspaces -m option to start mirroring for a dbspace, blobspace, or sbspace.

Syntax:

onspaces -m { *dbspace* | *blobspace* | *sbspace* } { | **-p***pathname* **-o***offset* **-m***pathnameoffset* | **-f***filename* } [**-y**]

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-f *filename*	Indicates that chunk-location information is in a file named *filename*	The file must be a buffered file that already exists. The path name must conform to the operating-system-specific rules for path names. For more information, see Using a File to Specify Chunk-Location Information with the -f Option on page 452.
-m	Adds mirroring for an existing dbspace, blobspace, or sbspace	User-data chunks in a mirrored sbspace need not be mirrored. The mirrored chunks should be on a different disk. You must mirror all the chunks at the same time.
-m *pathname offset*	The second time that *pathname* occurs in the syntax diagram, it indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that performs the mirroring. The second time *offset* appears in the syntax diagram, it indicates the offset to reach the mirrored chunk of the newly mirrored dbspace, blobspace, or sbspace. Also see the entries for *pathname* and *offset* in this table.	None.
-o *offset*	The first time that *offset* occurs in the syntax diagram, it indicates, in kilobytes, the offset into the disk partition or into the unbuffered device to reach the initial chunk	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 terabytes.

Element	Purpose	Key considerations
	of the newly mirrored dbspace, blobspace, or sbspace.	For more information, see allocating raw disk space on UNIX™, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-p *pathname*	The first time *pathname* occurs in the syntax diagram, it indicates the disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you want to mirror.	The chunk must be an existing unbuffered device or buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. For path name syntax, see your operating-system documentation.
-y	Causes the database server to automatically respond yes to all prompts	None.
blobspace	Names the blobspace that you want to mirror	Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*. For more information, see the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.
dbspace	Names the dbspace that you want to mirror	Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*. For background information, see the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.
sbspace	Names the sbspace that you want to mirror	Syntax must conform to the Identifier segment; see *HCL OneDB™ Guide to SQL: Syntax*. For background information, see the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.

Using a File to Specify Chunk-Location Information with the -f Option

You can create a file that contains the chunk-location information. Then, when you execute **onspaces**, use the **-f** option to indicate to the database server that this information is in a file whose name you specify in ***filename***.

The contents of the file should conform to the following format, with options separated by spaces and each set of primary and mirror chunks on separate lines:

```
primary_chunk_path offset mirror_chunk_path offset
```

If the dbspace that you are mirroring contains multiple chunks, you must specify a mirror chunk for each of the primary chunks in the dbspace that you want to mirror. For an example that enables mirroring for a multichunk dbspace, see starting mirroring for unmirrored dbspaces with **onspaces** in the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.

onspaces -r: Stop mirroring

Use the onspaces -r option to end mirroring for a dbspace, blobspace, or sbspace.

Syntax:

`onspaces -r` { *dbspace* | *blobspace* | *sbspace* } [`-y`]

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-r	Indicates to the database server that mirroring should be ended for an existing dbspace, blobspace, or sbspace	For background information, see the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.
-y	Causes the database server to respond yes to all prompts automatically	None.
blobspace	Names the blobspace for which you want to end mirroring.	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.
dbspace	Names the dbspace for which you want to end mirroring.	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.
sbspace	Names the sbspace for which you want to end mirroring	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For background information, see the chapter on using mirroring in the *HCL OneDB™ Administrator's Guide*.

onspaces -ren: Rename a dbspace, blobspace, sbspace, or extspace

Use the onspaces -ren option to rename a dbspace, blobspace, sbspace, or extspace.

Syntax:

`onspaces -ren` { *dbspace* | *blobspace* | *sbspace* | *extspace* } `-n`*name*

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-ren	Causes the database server to rename the specified blobspace, dbspace, extspace, or sbspace	**Restrictions**: You can rename a blobspace, dbspace, extspace, or sbspace when the database server is in quiescent mode. For more information, see the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-n *name*	Specifies the new name for the blobspace, dbspace, extspace, or sbspace	**Restrictions**: The blobspace, dbspace, external space, or sbspace name must be unique and cannot exceed 128 bytes. It must begin with a letter or underscore and must contain only letters, numbers, underscores, or the **$** character. For more information, see the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*. The syntax must conform to the Identifier segment. For more information, see the *HCL OneDB™ Guide to SQL: Syntax*.
blobspace	Names the blobspace to be renamed	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
dbspace	Names the dbspace to be renamed	**Restrictions:** You cannot rename a critical dbspace, such as the root dbspace or a dbspace that contains physical logs. **Additional Information:** If you rename dbspaces that are included in the DATASKIP list, update the DATASKIP configuration parameter with the new names using the onspaces -f command. Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
extspace	Names the extspace to be renamed	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
sbspace	Names the sbspace to be renamed	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see renaming spaces, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.

Renaming a dbspace, blobspace, sbspace, or extspace when Enterprise Replication is active

You can rename a space (dbspace, blobspace, sbspace, or extspace) when Enterprise Replication is active.

When you put the database server into quiescent mode to rename the space, Enterprise Replication will be disconnected. You can then rename the space. The servers will resynchronize after you put the database server into online mode.

If you want to rename the same space on another server, you must put that server into quiescent mode and rename the space separately. No enforced relationship is propagated between renamed spaces on different ER servers; the same tables can be in different spaces.

If the Enterprise Replication server also participates in High-Availability Data Replication (HDR), you can rename the dbspace on the primary server and it will be automatically propagate to the secondary server. (The secondary server cannot participate in Enterprise Replication.)

Performing an Archive after Renaming a Space

After renaming any space (except extspaces or temporary spaces), perform a level-0 archive of the renamed space and the root dbspace. This will ensure that you can restore the spaces to a state including or following the rename dbspace operation. It is also necessary prior to performing any other type of archive.

onspaces -s: Change status of a mirrored chunk

Use the onspaces -s option to change the status of a mirrored chunk in a dbspace, a non-primary chunk within a noncritical dbspace, a blobspace, or an sbspace.

Syntax:

onspaces -s { *dbspace* | *blobspace* | *sbspace* } **-p***pathname* **-o***offset* { **-D** | **-O** } [**-y**]

This command has an equivalent SQL administration API function.

Element	Purpose	Key considerations
-D	Indicates that you want to take the chunk down	None.
-o *offset*	Indicates, in kilobytes, the offset into the disk partition or unbuffered device to reach the chunk	**Restrictions**: Unsigned integer. The starting offset must be equal to or greater than 0. The starting offset plus the chunk size cannot exceed the maximum chunk size.The offset must be a multiple of the page size. The maximum offset is 4 terabytes.

Element	Purpose	Key considerations
		For more information, see allocating raw disk space on UNIX™, in the chapter on managing disk space in the *HCL OneDB™ Administrator's Guide*.
-O	Indicates that you want to restore the chunk and bring it online	None.
-p *pathname*	Indicates the disk partition or unbuffered device of the chunk	The chunk can be an unbuffered device or a buffered file. When you specify a path name, you can use either a full path name or a relative path name. However, if you use a relative path name, it must be relative to the directory that was the current directory when you initialized the database server. For path name syntax, see your operating-system documentation.
-s	Indicates that you want to change the status of a chunk	**Restrictions**: You can only change the status of a chunk in a mirrored pair or a non-primary chunk within a noncritical dbspace. For more information, see changing the mirror status in the *HCL OneDB™ Administrator's Guide*.
-y	Causes the database server to respond yes to all prompts automatically	None.
blobspace	Names the blobspace whose status you want to change	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see changing the mirror status in the *HCL OneDB™ Administrator's Guide*.
dbspace	Names the dbspace whose status you want to change	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For more information, see changing the mirror status in the *HCL OneDB™ Administrator's Guide*.
sbspace	Names the sbspace whose status you want to change	Syntax must conform to the Identifier segment; see the *HCL OneDB™ Guide to SQL: Syntax*. For background information, see changing the mirror status in the *HCL OneDB™ Administrator's Guide*.

Avoid overwriting a chunk

The chunks associated with each HCL OneDB™ instance are not known to other HCL OneDB™ instances. It is possible to inadvertently create a chunk on a file or device that is allocated as a chunk to another HCL OneDB™ instance, which results in data corruption.

If you attempt to initialize an instance, where the ROOTPATH configuration parameter specifies a file or device that is the root chunk of another instance, the command fails with the following message in the online.log:

```
DISK INITIALIZATION ABORTED: potential instance overwrite detected.
```

To disable this initialization check, set the FULL_DISK_INIT configuration parameter to 1 in your configuration file and try to initialize the instance again. However, this initialization check is restricted to the root chunk. Adding dbspaces or chunks succeeds even when the file or device is allocated to another instance.

The onstat utility

The onstat utility reads shared-memory structures and provides statistics about the database server at the time that the command runs.

You can combine multiple onstat option flags in a single command. The contents of shared memory might change as the onstat output displays. The onstat utility does not place any locks on shared memory, so running the utility does not affect performance.

You use SQL administration API commands that are equivalent to onstat commands.

onstat Portal: onstat Utility Commands Sorted by Functional Category

The information in this topic lists onstat commands that are sorted by functional category.

Each category represents a different HCL OneDB™ feature for which onstat commands are useful for providing troubleshooting and performance enhancement information. Commands that appear in bold typeface are especially useful for providing troubleshooting information. Certain onstat commands are specific to one category, while others provide more general information and are listed in more than one category.

Category List

Determine the appropriate category from the following list, then follow the link to the onstat options for that category.

- onstat Utility Archive Information Options on page 458
- onstat Utility Cache Information Options on page 458
- onstat Utility Compression Options on page 460
- onstat Utility Debugging Options on page 460
- onstat Utility Enterprise Replication Options on page 461
- onstat Utility High-Availability Replication Options on page 462
- onstat Utility I/O Options on page 464
- onstat Utility Locks and Latches Options on page 465
- onstat Utility Logs Options on page 465
- onstat Utility Memory Options on page 466
- onstat Utility Network Options on page 467
- onstat Utility Performance Checks (First Tier) on page 468
- onstat Utility Performance Checks (Second Tier) on page 469

onstat Utility Archive Information Options

Use the following onstat options to display information about archives and restores.

Table 135. onstat Utility Archive Information Options

Commands	Reference
onstat -D	Prints chunk I/O activity. Prints dbspace read/write activity for monitoring restore progress. onstat -D command: Print page-read and page-write information on page 506
onstat -g arc	Prints the last committed and any ongoing backups for each dbspace. onstat -g arc command: Print archive status on page 510

onstat Utility Cache Information Options

Use the following onstat options to display information about caches and cached data, including buffer pools.

Table 136. onstat Utility Cache Information Options

Commands	Reference
onstat -b	Prints buffer pages in use. onstat -b command: Print buffer information for buffers in use on page 485
onstat -B	Prints information about used buffers. onstat -B command: Prints information about used buffers on page 486
onstat -F	Prints state of buffer queue cleaners and I/O. onstat -F command: Print counts on page 507

Table 136. onstat Utility Cache Information Options (continued)

Commands	Reference
onstat -g cac	Prints summary and detailed information about all memory caches or about the specified cache. onstat -g cac command: Print information about caches on page 521
onstat -g dic	Prints data dictionary cache, containing system catalog data for tables. Prints one line of information for each table that is cached in the shared-memory dictionary. onstat -g dic command: Print table information on page 545
onstat -g dsc	Prints table distribution statistics for the optimizer. onstat -g dsc command: Print distribution cache information on page 553.
onstat -g prc	Prints the stored procedure (SPL) routine cache. Prints information about SPL routine cache. onstat -g prc command: Print sessions using UDR or SPL routines on page 606
onstat -g ssc	Prints the number of times that the database server reads the SQL statement in the cache. Displays the same output as onstat -g cac. For more information, see improving query performance in the *HCL OneDB™ Performance Guide*. onstat -g ssc command: Print SQL statement occurrences on page 659
onstat -g vpcache	Prints CPU virtual processor memory cache. onstat -g vpcache command: Print CPU virtual processor and tenant virtual processor private memory cache statistics on page 668
onstat -h	Prints buffer hash chain information. onstat -h command: Print buffer header hash chain information on page 677
onstat -O	Prints optical subsystem memory cache and staging-area (disk cache) blobspace for TEXT or BYTE data. onstat -O command: Print optical subsystem information on page 687

Table 136. onstat Utility Cache Information Options (continued)

Commands	Reference
onstat -p	Prints global (server) information regarding the effectiveness of buffer pool caching. onstat -p command: Print profile counts on page 688
onstat -X	Prints threads that are waiting for buffers. onstat -X command: Print thread information on page 712

onstat Utility Compression Options

Use the following onstat options to print compression information.

Table 137. onstat Utility Compression Options

Commands	Reference
onstat -g dsk	Prints progress of currently running compression operations. onstat -g dsk command: Print the progress of the currently running compression operation on page 555
onstat -g ppd	Prints partition compression dictionary information. onstat -g ppd command: Print partition compression dictionary information on page 602

onstat Utility Debugging Options

Use the following onstat options to display information that is useful for debugging problems with the server.

Table 138. onstat Utility Debugging Options

Commands	Reference
onstat -g dmp	Prints raw memory at a specified address for a number of given bytes. onstat -g dmp command: Print raw memory on page 549
onstat -g src	Searches for patterns in shared memory. Note that memory is byte-swapped on Intel™ platforms. onstat -g src command: Patterns in shared memory on page 658

Table 138. onstat Utility Debugging Options (continued)

Commands	Reference
onstat -o	Prints shared memory contents to a file. onstat -o command: Output shared memory contents to a file on page 686.

onstat Utility Enterprise Replication Options

Use the following onstat options to track Enterprise Replication statistics and to provide troubleshooting information. For additional information about Enterprise Replication see the cdr view and cdr view profile commands that are described in the *HCL OneDB™ Enterprise Replication Guide*.

Table 139. onstat Utility Enterprise Replication Options

Commands	Reference
onstat -g cat	Prints information from the Enterprise Replication global catalog. The global catalog contains a summary of information about the defined servers, replicates, and replicate sets on each of the servers within the enterprise. onstat -g cat: Print ER global catalog information on page
onstat -g cdr	Prints the output for all of the Enterprise Replication statistics commands. onstat -g cdr: Print ER statistics on page
onstat -g cdr config	Prints Enterprise Replication configuration parameters and environment variables. onstat -g cdr config: Print ER settings on page
onstat -g ddr	Prints status of Enterprise Replication components that read and process log records. onstat -g ddr: Print status of ER log reader on page
onstat -g dss	Prints activity of individual data sync (transaction processing) threads. onstat -g dss: Print statistics for data sync threads on page
onstat -g dtc	Prints delete table cleaner activity. Deleted or updated rows that are placed in the delete table are purged at intervals. onstat -g dtc: Print statistics about delete table cleaner on page

Table 139. onstat Utility Enterprise Replication Options (continued)

Commands	Reference
onstat -g grp	Prints Enterprise Replication grouper statistics. The grouper evaluates the log records, rebuilds the individual log records into the original transaction, packages the transaction, and queues the transaction for transmission. onstat -g grp: Print grouper statistics on page
onstat -g nif	Prints network interface statistics. Shows the state of the network interface, servers, and data transfer among servers. onstat -g nif: Print statistics about the network interface on page
onstat -g que	Prints statistics for the high-level queue interface (which is common to all of the queues of the Enterprise Replication Queue Manager). onstat -g que: Print statistics for all ER queues on page
onstat -g rcv	Prints receive manager statistics. onstat -g rcv: Print statistics about the receive manager on page
onstat -g rep	Prints events that are in the queue for the schedule manager. onstat -g rep: Prints the schedule manager queue on page
onstat -g rqm	Prints statistics and contents of the low-level queues (send queue, receive queue, ack send queue, sync send queue, and control send queue) managed by the Reliable Queue Manager (RQM). onstat -g rqm: Prints statistics for RQM queues on page
onstat -g sync	Prints synchronization status. onstat -g sync: Print statistics about synchronization on page

onstat Utility High-Availability Replication Options

Use the following onstat options to monitor high-availability cluster environments and the Connection Manager.

Table 140. onstat Utility High-Availability Replication Options

Commands	Reference
onstat -g cluster	Prints high-availability cluster information. onstat -g cluster command: Print high-availability cluster information on page 532
onstat -g cmsm	Prints Connection Manager information. onstat -g cmsm command: Print Connection Manager information on page 537
onstat -g dri	Prints data-replication information. See *Monitoring High-Availability Data-Replication status* in the *HCL OneDB™ Administrator's Guide*. onstat -g dri command: Print high-availability data replication information on page 550.
onstat -g ipl	Prints index page logging status. onstat -g ipl command: Print index page logging status information on page 574
onstat -g laq	Prints information about log recovery apply queues. onstat -g laq command: Print log apply queues on page 578
onstat -g proxy	Prints proxy distributors for high-availability. onstat -g proxy command: Print proxy distributor information on page 608
onstat -g rss	Prints remote stand-alone server (RSS) information. onstat -g rss command: Print RS secondary server information on page 619
onstat -g sds	Prints shared disk secondary (SDS) server information. onstat -g sds command: Print SD secondary server information on page 629
onstat -g smx	Prints Server Multiplexer Group (SMX) connections in high-availability environments. Prints data transfer statistics and encryption status.Prints data transfer statistics. onstat -g smx command: Print multiplexer group information on page 651

onstat Utility I/O Options

Use the following onstat options to track input and output (read and write) activity.

Table 141. onstat Utility I/O Options

Commands	Reference
onstat -D	Prints chunk I/O activity. onstat -D command: Print page-read and page-write information on page 506
onstat -g cpu	Prints runtime statistics for each thread. onstat -g cpu: Print runtime statistics on page 541
onstat -g ioa	Prints combined information from onstat -g ioq (queues), onstat -g iov (virtual processors), and onstat -g iob (big buffer). onstat -g ioa command: Print combined onstat -g information on page 568
onstat -g iob	Prints the big buffer usage summary. onstat -g iob command: Print big buffer use summary on page 570
onstat -g iof	Prints I/O statistics by file or chunk. This option is similar to the onstat -D option, but also displays information about non-chunk, temporary, and sort-work files. onstat -g iof command: Print asynchronous I/O statistics on page 571
onstat -g iog	Prints AIO global information. onstat -g iog command: Print AIO global information on page 572
onstat -g ioq	Prints queue read/write statistics and queue length. onstat -g ioq command: Print I/O queue information on page 572. Also see the *HCL OneDB™ Performance Guide*.
onstat -g iov	Prints asynchronous I/O statistics by virtual processor. onstat -g iov command: Print AIO VP statistics on page 575
onstat -p	Prints global disk activity, including sequential scans.

Table 141. onstat Utility I/O Options (continued)

Commands	Reference
	onstat -p command: Print profile counts on page 688

onstat Utility Locks and Latches Options

Use the following onstat options to display information about locks.

Table 142. onstat Utility Locks and Latches Options

Commands	Reference
onstat -k	Prints information about active locks. onstat -k command: Print active lock information on page 679
onstat -L	Prints the number of locks on a lock free list. onstat -L command: Print the number of free locks on page 685
onstat -p	Prints global statistics on lock requests, lock waits, and latch waits. onstat -p command: Print profile counts on page 688
onstat -s	Prints latch (mutex) information. onstat -s command: Print latch information on page 700

onstat Utility Logs Options

Use the following onstat options to monitor logical and physical logs.

Table 143. onstat Utility Logs Options

Commands	Reference
onstat -g ipl	Prints index page logging information in high-availability environments. onstat -g ipl command: Print index page logging status information on page 574
onstat -l	Prints status of physical and logical logs, and log buffering. onstat -l command: Print physical and logical log information on page 681

onstat Utility Memory Options

Use the following onstat options to monitor the various aspects of server memory allocation and use.

Table 144. onstat Utility Memory Options

Commands	Reference
onstat -g afr	Prints allocated memory fragments for a specified session or shared-memory pool. To obtain the pool name, see the onstat -g mem option. onstat -g afr command: Print allocated memory fragments on page 509
onstat -g ffr (*pool name session ID*)	Prints free fragments for a session or shared memory pool. onstat -g ffr command: Print free fragments on page 559
onstat -g lmm	Prints information about automatic low memory management settings and recent activity: onstat -g lmm command: Print low memory management information on page 580
onstat -g mem	Prints session or pool virtual shared memory statistics. onstat -g mem command: Print pool memory statistics on page 585
onstat -g mgm	Prints Memory Grant Manager (parallel and sort operations) resource information. onstat -g mgm command: Print MGM resource information on page 586. Also see the *HCL OneDB™ Performance Guide*.
onstat -g nbm	Prints block map for non-resident segments. onstat -g nbm command: Print a block bit map on page 590
onstat -g rbm	Prints block map for resident segment. onstat -g rbm command: Print a block map of shared memory on page 618
onstat -g seg	Prints memory segment statistics. onstat -g seg command: Print shared memory segment statistics on page 634. Also see the *HCL OneDB™ Administrator's Guide*.

Table 144. onstat Utility Memory Options (continued)

Commands	Reference
onstat -g ses	Prints session information, including memory breakdown. For detailed information, use: onstat -g ses *session_id* onstat -g ses command: Print session-related information on page 636 Also see the *HCL OneDB™ Performance Guide*
onstat -g stm	Prints SQL statement memory use. onstat -g stm command: Print SQL statement memory usage on page 662
onstat -g stq	Prints stream queue buffers. onstat -g stq command: Print queue information on page 663
onstat -g ufr	Prints memory pool fragments for a session or shared memory pool in use. onstat -g ufr command: Print memory pool fragments on page 667
onstat -R	Prints buffer pool queues and their status. onstat -R command: Print LRU, FLRU, and MLRU queue information on page 697

onstat Utility Network Options

Use the following onstat options to monitor shared memory and network connection services.

Table 145. onstat Utility Network Options

Commands	Reference
onstat -g nsc	Prints shared-memory status by *client id*. If *client id* is omitted, all client status areas are displayed. This command prints the same status data as the nss command. onstat -g nsc command: Print current shared memory connection information on page 591
onstat -g nsd	Prints network shared-memory data for poll threads. onstat -g nsd command: Print poll threads shared-memory data on page 594

Table 145. onstat Utility Network Options (continued)

Commands	Reference
onstat -g nss	Prints network shared-memory status by *session id*. If *session id* is omitted, all session status areas are displayed. This command prints the same status data as the onstat -g nsc command. onstat -g nss command: Print shared memory network connections status on page 594
onstat -g ntd	Prints network statistics by service. onstat -g ntd command: Print network statistics on page 595
onstat -g ntm	Prints network mail statistics. onstat -g ntm command: Print network mail statistics on page 596
onstat -g ntt	Prints network user times. onstat -g ntt command: Print network user times on page 597
onstat -g ntu	Prints network user statistics. onstat -g ntu command: Print network user statistics on page 597

onstat Utility Performance Checks (First Tier)

Use the following onstat options to monitor performance and to check for performance impediments. Use the second-tier onstat options (and other onstat commands) to further narrow the problem.

Table 146. onstat Utility Performance Checks (First Tier)

Commands	Reference
onstat -c	Prints server configuration. onstat -c command: Print ONCONFIG file contents on page 489
onstat -D	Prints chunk I/O. onstat -D command: Print page-read and page-write information on page 506
onstat -g ath	Prints status and statistics for all threads. The sqlexec thread is a client session thread. The rstcb value corresponds to the user field of the onstat -u command.

Table 146. onstat Utility Performance Checks (First Tier) (continued)

Commands	Reference
	onstat -g ath command: Print information about all threads on page 512. For information about using onstat -g ath to print Enterprise Replication threads, see the *HCL OneDB™ Enterprise Replication Guide*.
onstat -g ckp	Prints checkpoint history and display configuration recommendations. onstat -g ckp command: Print checkpoint history and configuration recommendations on page 524
onstat -g cpu	Prints runtime statistics for each thread. onstat -g cpu: Print runtime statistics on page 541
onstat -g ioq	Prints pending I/O operations for the *queue name*. onstat -g ioq command: Print I/O queue information on page 572
onstat -p	Prints global server performance profile. onstat -p command: Print profile counts on page 688
onstat -u	Prints status and statistics for user threads. If a thread is waiting for a resource, this command identifies the type (flags field) and address (wait field) of the resource. onstat -u command: Print user activity profile on page 704

onstat Utility Performance Checks (Second Tier)

Use the following onstat options to identify performance impediments.

Table 147. onstat Utility Performance Checks (Second Tier)

Commands	Reference
onstat -b	Prints active buffers. onstat -b command: Print buffer information for buffers in use on page 485
onstat -g act	Prints active threads. onstat -g act command: Print active threads on page 509

Table 147. onstat Utility Performance Checks (Second Tier) (continued)

Commands	Reference
onstat -g glo	Prints virtual processors and their operating system processes (oninit processes). Prints virtual processor CPU use. On Windows™, the virtual processors are operating system threads, and the values in the pid field are thread IDs. onstat -g glo command: Print global multithreading information on page 560
onstat -g mgm	Prints Memory Grant Manager resource information. onstat -g mgm command: Print MGM resource information on page 586
onstat -g rah	Prints read-ahead request information onstat -g rah command: Print read-ahead request statistics on page 615
onstat -g rea	Prints threads in the ready queue that are waiting for CPU resources. onstat -g rea command: Print ready threads on page 619
onstat -g seg	Prints shared-memory-segment statistics. This option shows the number and size of shared-memory segments that are allocated to the database server. onstat -g seg command: Print shared memory segment statistics on page 634.
onstat -g wai	Prints waiting threads; all threads that are waiting for mutex or condition, or yielding. onstat -g wai command: Print wait queue thread list on page 670
onstat -k	Prints active locks. onstat -k command: Print active lock information on page 679

onstat Utility Table Options

Use the following onstat options to display information about table status and table statistics.

Table 148. onstat Utility Table Options

Commands	Reference
onstat -g buf	Prints buffer pool profile information.

Table 148. onstat Utility Table Options (continued)

Commands	Reference
	onstat -g buf command: Print buffer pool profile information on page 516
onstat -g lap	Prints information about the status of currently active light appends (writes bypassing the buffer pool). onstat -g lap command: Print light appends status information on page 577
onstat -g opn	Prints open partitions (tables). onstat -g opn command: Print open partitions on page 598
onstat -g ppf	Prints partition profile (activity data) for the specified partition number or prints profiles for all partitions. onstat -g ppf command: Print partition profiles on page 603
onstat -g scn	Prints information about the progress of a scan, based on rows scanned on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data, and identifies whether a scan is a light or bufferpool scan. onstat -g scn command: Print scan information on page 626
onstat -P	Prints table and B-tree pages in the buffer pool, listed by partition (table). onstat -P command: Print partition information on page 693
onstat -t onstat -T	Prints basic tblspace (partition) information for active (t) or all (T) tblspaces. onstat -t and onstat -T commands: Print tblspace information on page 702

onstat Utility Thread Options

Use the following onstat options to display the status and activity of threads.

Table 149. onstat Utility Thread Options

Commands	Reference
onstat -g act	Prints active threads. This output is included in onstat -g ath output.

Table 149. onstat Utility Thread Options (continued)

Commands	Reference
	onstat -g act command: Print active threads on page 509
onstat -g ath	Prints all threads. onstat -g ath command: Print information about all threads on page 512. For information about using onstat -g ath to print Enterprise Replication threads, see the *HCL OneDB™ Enterprise Replication Guide*.
onstat -g bth	Displays the dependencies between blocking and waiting threads. onstat -g bth and -g BTH: Print blocked and waiting threads on page 513
onstat -g BTH	Displays session and stack information for the blocking threads. onstat -g bth and -g BTH: Print blocked and waiting threads on page 513
onstat -g cpu	Prints runtime statistics for each thread. onstat -g cpu: Print runtime statistics on page 541
onstat -g rea	Prints ready threads (threads that are waiting for CPU resources). This output is included in the onstat -g ath output. onstat -g rea command: Print ready threads on page 619.
onstat -g sle	Prints information about threads that are sleeping for a specified time. Does not include threads that are sleeping forever. onstat -g sle command: Print all sleeping threads on page 648
onstat -g stk	Prints the stack of a specified thread or prints stacks for all threads. onstat -g stk command: Print thread stack on page 661
onstat -g sts	Prints maximum and current stack use per thread. onstat -g sts command: Print stack usage for each thread on page 663
onstat -g tpf	Prints thread activity statistics. onstat -g tpf command: Print thread profiles on page 665

Table 149. onstat Utility Thread Options (continued)

Commands	Reference
onstat -g wai	Prints waiting (idle, sleeping, and waiting) threads. Included in onstat -g ath output. onstat -g wai command: Print wait queue thread list on page 670
onstat -g wst	Prints wait statistics for threads. onstat -g wst command: Print wait statistics for threads on page 672

onstat Utility User/Session Options

Use the following onstat options to display information about the user environment and active sessions.

Table 150. onstat Utility User/Session Options

Commands	Reference
onstat -g env	Prints the values of environment variables the database server is using. onstat -g env command: Print environment variable values on page 557
onstat -g his	Prints SQL tracing information. onstat -g his command: Print SQL trace information on page 563
onstat -g pqs	Prints operators that are used in currently running SQL queries. onstat -g pqs command: Print operators for all SQL queries on page 605
onstat -g ses	Prints summary information for all active sessions or detailed information for individual sessions. onstat -g ses command: Print session-related information on page 636
onstat -g spf	Prints prepared statement profiles for all active sessions. onstat -g spf: Print prepared statement profiles on page 657
onstat -g sql	Prints SQL information for all active sessions or detailed SQL information for individual sessions. onstat -g sql command: Print SQL-related session information on page 655

Table 150. onstat Utility User/Session Options (continued)

Commands	Reference
onstat -G	Prints global transactions. onstat -G command: Print TP/XA transaction information on page 674
onstat -u	Prints status of user threads and their global read/write statistics. onstat -u command: Print user activity profile on page 704
onstat -x	Prints information about transactions. onstat -x command: Print database server transaction information on page 708

onstat Utility Virtual Processor Options

Use the following onstat options to display information and statistics for virtual processors.

Table 151. onstat Utility Virtual Processor Options

Commands	Reference
onstat -g glo	Prints global multithreading information and global statistics for virtual processor classes and individual virtual processors. On Windows™, the virtual processors are operating system threads, and the values in the pid field are thread IDs. onstat -g glo command: Print global multithreading information on page 560
onstat -g sch	Prints the number of semaphore operations, spins, and busy waits for each virtual processor. On Windows™, the virtual processors are operating system threads, and the values in the pid field are thread IDs. onstat -g sch command: Print VP information on page 625

onstat Utility Waiting Options

Use the following onstat options to display information about wait conditions for threads.

Table 152. onstat Utility Waiting Options

Commands	Reference
onstat -g con	Prints IDs of threads that are waiting for conditions.

Table 152. onstat Utility Waiting Options (continued)

Commands	Reference
	onstat -g ath to print thread information. See onstat -g con command: Print condition and thread information on page 540
onstat -g lmx	Prints all locked mutexes. onstat -g lmx command: Print all locked mutexes on page 582
onstat -g qst	Prints queue-wait statistics for mutex and condition queues. onstat -g qst command: Print wait options for mutex and condition queues on page 614
onstat -g rwm	Prints read/write mutexes. onstat -g rwm command: Print read and write mutexes on page 624
onstat -g spi	Prints spin locks with long spins and spin lock statistics. onstat -g spi command: Print spin locks with long spins on page 653
onstat -g wai	Prints waiting threads; all threads that are waiting for mutex or condition, or yielding. onstat -g wai command: Print wait queue thread list on page 670
onstat -g wmx	Prints all mutexes with waiters. onstat -g wmx command: Print all mutexes with waiters on page 671

Other Useful onstat Utility Options

Table 153. Other Useful onstat Utility Options

Commands	Reference
onstat -	Prints onstat header; includes engine version, status (online, Quiescent, and so on), elapsed time since initialization, and memory footprint. onstat - command: Print output header on page 483
onstat	Prints onstat usage options. onstat -- command: Print onstat options and functions on page 484

Table 153. Other Useful onstat Utility Options (continued)

Commands	Reference
onstat *options infile*	Print onstat output using a shared memory dump (infile) as input. Running onstat Commands on a Shared Memory Dump File on page 484
onstat -a	Prints collective onstat outputs. onstat -a command: Print overall status of the database server on page 485
onstat -c	Prints the server configuration file. onstat -c command: Print ONCONFIG file contents on page 489
onstat -C	Prints B-tree index scanner information (shows statistics about index cleaning). onstat -C command: Print B-tree scanner information on page 489
onstat -d	Prints chunk information. onstat -d command: Print chunk information on page 498
onstat -f	Prints dbspaces configured for dataskip. onstat -f command: Print dbspace information affected by dataskip on page 507
onstat -g all	Prints diagnostic information. onstat -g all command: Print diagnostic information on page 510
onstat -g cfg	Prints a list of configuration parameters with their current values. onstat -g cfg command: Print the current values of configuration parameters on page 529
onstat -g dbc	Prints statistics about dbScheduler and dbWorker threads. onstat -g dbc command: Print dbScheduler and dbWorker thread statistics on page 542
onstat -g dis	Prints a list of database servers, their status, directory location, configuration information, and host name.

Table 153. Other Useful onstat Utility Options (continued)

Commands	Reference
	onstat -g dis command: Print database server information on page 546
onstat -g dll	Prints a list of dynamic libraries that are loaded. onstat -g dll command: Print dynamic link library file list on page 548
onstat -g osi	Prints information about operating system resources and parameters. onstat -g osi: Print operating system information on page 598
onstat -g pos	Prints values from `$ONEDB_HOME/etc/.infos.servernum` file, which are used by clients such as onmode for shared memory connections to the server. onmode -R rebuilds the `$ONEDB_HOME/etc/.infos.servernum` file. onstat -g pos command: Print file values on page 602
onstat -g smb	Prints detailed information about sbspaces. onstat -g smb command: Print sbspaces information on page 649
onstat -g sym	Prints symbol table information for the oninit utility. onstat -g sym command: Print symbol table information for the oninit utility on page 664
onstat -i	Changes onstat mode to interactive. onstat -i command: Initiate interactive mode on page 678
onstat -m	Prints message log contents. onstat -m command: Print recent system message log information on page 685
onstat -O	Prints Optical subsystem cache information. onstat -O command: Print optical subsystem information on page 687
onstat -r	Prints repetitive onstat execution. onstat -r command: Repeatedly print selected statistics on page 695

Table 153. Other Useful onstat Utility Options (continued)

Commands	Reference
onstat -z	Resets the accumulated statistics to zero. onstat -z command: Clear statistics on page 715

Monitor the database server status

To monitor the database server status, view the heading of the onstat command.

Whenever the database server is blocked, onstat displays the following line after the banner line:

```
Blocked: reason
```

The variable *reason* can be one or more of the following values.

Reason	Description
ADMINISTRATION	Database is in administration mode
ARCHIVE	Ongoing storage-space backup
ARCHIVE_EBR	Blocked for External Backup and Recovery.
CHG_PLOG	Blocked while physical log is being changed.
CKPT	Checkpoint
CKPT INP	Interval checkpoint in progress
DBS_DROP	Dropping a dbspace
DDR	Discrete data replication
DYNAMIC_LOG	Log file is being added dynamically
DYNAMIC_LOG_FOR_ER	Log file is being added dynamically in ER setup
FREE_LOG	Log file is being freed
HA_CONV_STD	Blocked while High Availability server is being converted to standard server.
HA_FAILOVER	Blocked while High Availability server failover being processed.
HANG_SYSTEM	Database server failure
LAST_LOG_RESERVED4BACKUP	Waiting for last available log to be backed up
LBU	Logs full high-watermark
LOG_DROP	Log file is being dropped

Reason	Description
LONGTX	Long transaction
MEDIA_FAILURE	Media failure
OVERRIDE_DOWN_SPACE	Waiting to override down dbspace setting because the ONDBSPACEDOWN onconfig parameter is set to WAIT

In this table, the value CHKP INP does not indicate that the database server is blocked, but that a nonblocking interval checkpoint is in progress while the buffer pool is being flushed. This CHKP INP value appears in the status line of onstat output until all pages in the shared-memory buffer pool have been written to disk. For information about setting interval checkpoints to flush the buffer pool, see the CKPTINTVL configuration parameter on page 61.

onstat command syntax

The complete syntax for the onstat command, including information about the interactive mode and how to have options to execute repeatedly.

```
onstat [ < -FILE option > (explicit id ) ] -pu { [ { | -a | -b | -B | -c | -C | -d | -D | -f | -F | -g monitoring_option | -G | -h
| -i | -k | -l | -m | -o [ { nobuffs | full } ] { [outfile] } | -O | -p | -P | -r [seconds] | -R | -s | -t | -T | -u | -x | -X | -
z } ] [ infile ] (explicit id )  | -  | --  | -V | -version }
```

Element	Purpose	Key Considerations
-	Displays the output header only.	See onstat - command: Print output header on page 483.
--	Displays a listing of all onstat options and their functions	See onstat -- command: Print onstat options and functions on page 484. This option cannot be combined with any other onstat option.
-a	Interpreted as onstat -cuskbtdlp. Displays output in that order.	See onstat -a command: Print overall status of the database server on page 485.
-b	Displays information about buffers currently in use, including number of resident pages in the buffer pool	See onstat -b command: Print buffer information for buffers in use on page 485.
-B	Obtains information about all database server buffers, not just buffers currently in use.	See onstat -B command: Prints information about used buffers on page 486.
-c	Displays the ONCONFIG file:	See onstat -c command: Print ONCONFIG file contents on page 489.

Element	Purpose	Key Considerations
	• `$ONEDB_HOME/etc/$ONCONFIG` for UNIX™ • `%ONEDB_HOME%\etc\%ONCONFIG%` for Windows™	
-C	Prints B-tree scanner information	See onstat -C command: Print B-tree scanner information on page 489.
-d	Displays information for chunks in each storage space	See onstat -d command: Print chunk information on page 498.
-D	Displays page-read and page-write information for the first 50 chunks in each dbspace	See onstat -D command: Print page-read and page-write information on page 506.
-f	Lists the dbspaces currently affected by the DATASKIP feature	See onstat -f command: Print dbspace information affected by dataskip on page 507.
-F	Displays a count for each type of write that flushes pages to disk	See onstat -F command: Print counts on page 507.
-g ***option***	Prints monitoring option	See onstat -g monitoring options on page 509.
-G	Prints global transaction IDs	See onstat -G command: Print TP/XA transaction information on page 674.
-h	Provides information on the buffer header hash chains	See onstat -h command: Print buffer header hash chain information on page 677.
-i	Puts the onstat utility into interactive mode	See onstat -i command: Initiate interactive mode on page 678.
-k	Displays information about active locks	See onstat -k command: Print active lock information on page 679.
-l	Displays information about physical and logical logs, including page addresses	See onstat -l command: Print physical and logical log information on page 681.
-m	Displays the 20 most recent lines of the database server message log	Output from this option lists the full pathname of the message-log file and the 20 file entries. A date-and-time header separates the entries for each day. A time stamp prefaces single entries within each day. The name of the message log is specified as MSGPATH in the ONCONFIG file.

Element	Purpose	Key Considerations
		See onstat -m command: Print recent system message log information on page 685.
-o	Saves a copy of the shared-memory segments to *outfile*	See onstat -o command: Output shared memory contents to a file on page 686.
-O	Displays information about the Optical Subsystem memory cache and staging-area blobspace	See onstat -O command: Print optical subsystem information on page 687.
-p	Displays profile counts.	See onstat -p command: Print profile counts on page 688.
-P	Displays for all partitions the partition number and the break-up of the buffer-pool pages that belong to the partition	See onstat -P command: Print partition information on page 693.
-pu	If you invoke onstat without any options, the command is interpreted as onstat -pu (-p option and -u option). Displays profile counts and prints a profile of user activity	See onstat -p command: Print profile counts on page 688 and onstat -u command: Print user activity profile on page 704.
-r ***seconds***	Repeats the accompanying onstat options after a wait time specified in *seconds* between each execution	See onstat -r command: Repeatedly print selected statistics on page 695.
-R	Displays detailed information about the LRU queues, FLRU queues, and MLRU queues	See onstat -R command: Print LRU, FLRU, and MLRU queue information on page 697.
-s	Displays general latch information	See onstat -s command: Print latch information on page 700.
-t	Displays tblspace information, including residency state, for active tblspaces	See onstat -t and onstat -T commands: Print tblspace information on page 702.
-T	Displays tblspace information for all tblspaces	See onstat -t and onstat -T commands: Print tblspace information on page 702.
-u	Prints a profile of user activity	See onstat -u command: Print user activity profile on page 704.
-V	Displays the software version number and the serial number. This option cannot be combined with any other onstat option.	See Obtaining utility version information on page 312.

Element	Purpose	Key Considerations
-version	Displays the build version, host, OS, number and date, as well as the GLS version. This option cannot be combined with any other onstat option.	See Obtaining utility version information on page 312.
-x	Displays information about transactions	See onstat -x command: Print database server transaction information on page 708.
-X	Obtains precise information about the threads that are sharing and waiting for buffers	See onstat -X command: Print thread information on page 712.
-z	Sets the profile counts to 0	See onstat -z command: Clear statistics on page 715.
infile	Specifies a source file for the onstat command	This file must include a previously stored shared-memory segment that you created with the onstat -o command. For instructions on how to create the *infile* with onstat -o, see onstat -o command: Output shared memory contents to a file on page 686. For information about running onstat on the source file, see Running onstat Commands on a Shared Memory Dump File on page 484.

Interactive execution

To put the onstat utility in interactive mode, use the -i option. Interactive mode allows you to enter multiple options, one after the other, without exiting the program. For information on using interactive mode, see onstat -i command: Initiate interactive mode on page 678.

Continuous onstat command execution

Use the onstat -r option combined with other onstat options to cause the other options to execute repeatedly at a specified interval. For information, see onstat -r command: Repeatedly print selected statistics on page 695.

onstat command: Equivalent to the onstat -pu command

If you invoke onstat without any options, the command is interpreted as onstat -pu (the -p option and the -u option).

Syntax:

```
onstat
```

onstat - command: Print output header

All onstat output includes a header. The onstat - command displays only the output header and the value that is returned from this command indicates the database server mode.

Syntax:

```
onstat  -
```

The header takes the following form:

Version--*Mode (Type)*--*(Checkpnt)*--`Up` *Uptime*--*Sh_mem* `Kbytes`

Version

Is the product name and version number

Mode

Is the current operating mode.

(*Type*)

If the database server uses High-Availability Data Replication, indicates whether the type is primary or secondary

If the database server is not involved in data replication, this field does not appear. If the type is primary, the value `P` appears. If the type is secondary, the value `S` appears.

(Checkpnt)

Is a checkpoint flag

If it is set, the header might display two other fields after the mode if the timing is appropriate:

(CKPT REQ)

Indicates that a user thread has requested a checkpoint

(CKPT INP)

Indicates that a checkpoint is in progress. During the checkpoint, access is limited to read only. The database server cannot write or update data until the checkpoint ends

Uptime

Indicates how long the database server has been running

If the system time is manually changed to the past and the server startup time is later than the current system time, the uptime is not available. In this situation, the header displays the text `Uptime Unavailable`.

Sh_mem

Is the size of database server shared memory, expressed in kilobytes

A sample header for the database server follows:

```
OneDB--On-Line--Up 15:11:41--9216 Kbytes
```

If the database server is blocked, the onstat header output includes an extra line. For information about status codes in that line, see Monitor the database server status on page 478.

Return codes

When you exit the onstat utility, there are several useful codes that are displayed. See Return codes on exiting the onstat utility on page 716.

onstat -- command: Print onstat options and functions

Use the onstat -- command to display a listing of all of the onstat options and their functions. You cannot combine this option with any other flag.

Syntax:

`onstat --`

Running onstat Commands on a Shared Memory Dump File

You can run onstat commands against a shared memory dump file. The shared memory dump file can be produced explicitly by using the **onstat -o** command. If the DUMPSHMEM configuration parameter is set to 1 or set to 2, the dump file is created automatically at the time of an assertion failure.

Syntax:

`onstat` *optionsinfile*

When using the command line, enter the source file as the final argument. The following example prints information about all threads for the shared memory dump contained in the file named `onstat.out`, rather than attempting to attach to the shared memory of a running server.

```
onstat -g ath onstat.out
```

For instructions on how to create the memory dump file with onstat -o, see onstat -o command: Output shared memory contents to a file on page 686.

Running onstat Commands on a Shared Memory Dump File Interactively

Use onstat -i (interactive mode) to run more than one onstat command against a dump file. Interactive mode can save time because the file is read only once. In command-line mode, each command reads the file.

The following example reads the shared memory dump file and enters interactive mode. Other onstat commands can be executed against the dump file in the normal interactive fashion.

```
onstat -i source_file
```

For information about interactive mode, see onstat -i command: Initiate interactive mode on page 678.

Running onstat Commands on a Shared Memory Dump File Created Without a Buffer Pool

Certain onstat commands have different output when you run them on a dump file created without the buffer pool (created with onstat -o nobuffs or with the DUMPSHMEM configuration parameter set to 2):

- If you run onstat -B on a dump file created without the buffer pool, the output will display `0` in the `memaddr`, `nslots`, and `pgflgs` columns.
- If you run onstat -g seg on a dump file created without the buffer pool, the output will show both the original and nobuffs resident segment size.
- If you run onstat -P on a shared-memory dump file that does not have the buffer pool, the output is:

```
Nobuffs dumpfile -- this information is not available
```

onstat -a command: Print overall status of the database server

Use the onstat -a command to display information about the status of the database server. This command does not display information about all of the onstat options, only about those onstat options used for initial troubleshooting.

Syntax:

```
onstat  -a
```

onstat -b command: Print buffer information for buffers in use

Use the onstat -b option to display information about the buffers that are currently in use, including the total number of resident pages in the buffer pool.

Syntax:

```
onstat  -b
```

The maximum number of buffers available is specified in the **buffers** field in the BUFFERPOOL configuration parameter in the `ONCONFIG` file.

The onstat -b command also provides summary information about the number of modified buffers, the total number of resident pages in the buffer pool, the total number of buffers available, the number of hash buckets available, and the size of the buffer in bytes (the page size).

```
123 modified, 23 resident, 2000 total, 2048 hash buckets, 2048 buffer size.
```

For information about displaying information about all buffers, use onstat -B command: Prints information about used buffers on page 486.

Example output

Following is sample output from the onstat -b command. For a description of the output, see onstat -B command: Prints information about used buffers on page 486.

Figure 55. onstat -b command output

```
Buffer pool page size: 4096

address          userthread flgs pagenum memaddr          nslots pgflgs xflgs owner waitlist
70000001097e9e8 0           c07 1:47841 7000000118e0000 10     1      0     0     0
700000010982188 0           807 1:47827 700000011939000 225    90     10    0     0
2011 modified, 50000 total, 65536 hash buckets, 4096 buffer size
```

onstat -B command: Prints information about used buffers

Use the onstat -B option to display information about buffers that are not on the free-list.

Syntax:

onstat -B

Both onstat -B and onstat -b display the similar information, except that the onstat -b command only displays buffers that are currently being accessed by a user thread. The onstat -B command displays information for all the buffers that are not on the free-list.

For information about running the onstat -B command on a dump file created without the buffer pool, see Running onstat Commands on a Shared Memory Dump File on page 484.

Example output

Figure 57. onstat -B command output

```
Buffer pool page size: 4096

address          userthread flgs pagenum memaddr          nslots pgflgs xflgs owner            waitlist
700000010932fe8 0           806  1:40264 700000011150000 5      8b0    0     0                0
700000010933088 0           806  1:40284 700000011151000 5      870    0     0                0
.
.
700000010933e48 0           806  1:44585 700000011167000 113    8890   0     0                0
700000010933ee8 0           806  1:42578 700000011168000 5      802    0     0                0
700000010933f88 0           86   1:44348 700000011169000 39     890    0     ffffffffffffffff 0
700000010934028 0           6    1:8     70000001116a000 0      1800   0     0                0
7000000109340c8 0           806  1:246   70000001116b000 5      802    0     0                0
.
.
700000010941c28 0           807  1:46486 7000000112ca000 123    801    0     0                0
700000010941ea8 0           806  1:47101 7000000112ce000 10     801    0     0                0
 25 modified, 50000 total, 65536 hash buckets, 4096 buffer size
```

Output description

Buffer pool page size

the size of the buffer pool pages in bytes

address

the address of the buffer header in the buffer table

userthread

the address of the most recent user thread to access the buffer table. Many user threads might be reading the same buffer concurrently.

flgs

Uses the following flag bits to describe the buffer:

0x01

Modified data

0x02

Data

0x04

LRU

0x08

Error

pagenum

the physical page number on the disk

memaddr

the buffer memory address

nslots

the number of slot-table entries in the page

This field indicates the number of rows (or portions of a row) that are stored on the page.

pgflgs

Uses the following values, alone or in combination, to describe the page type:

1

Data page

2

Tblspace page

4

Free-list page

8

Chunk free-list page

9

Remainder data page

b

Partition resident blobpage

c

Blobspace resident blobpage

d

Blob chunk free-list bit page

e

Blob chunk blob map page

10

B-tree node page

20

B-tree root-node page

40

B-tree branch-node page

80

B-tree leaf-node page

100

Logical-log page

200

Last page of logical log

400

Sync page of logical log

800

Physical log

1000

Reserved root page

2000

No physical log required

8000

B-tree leaf with default flags

xflgs

Uses the following flag bits to describe buffer access:

0x10

share lock

0x80

exclusive lock

owner

the user thread that set the **xflgs** buffer flag

waitlist

the address of the first user thread that is waiting for access to this buffer

For a complete list of all threads waiting for the buffer, refer to onstat -X command: Print thread information on page 712.

onstat -c command: Print ONCONFIG file contents

Use the onstat -c command to display the contents of the ONCONFIG file.

```
>>-onstat-- -c-------------------------------------------------><
```

The database server first checks if you have assigned a value to the environment variable **ONCONFIG**. You can use the onstat -c option with the database server in any mode, including offline.

UNIX Only:

On UNIX™, if you have set **ONCONFIG**, onstat -c displays the contents of the **$ONEDB_HOME/etc/$ONCONFIG** file. If not, by default, onstat -c displays the contents of **$ONEDB_HOME/etc/onconfig**.

Windows Only:

On Windows™, if you have set **ONCONFIG**, onstat -c displays the contents of the **%ONEDB_HOME%\etc\%ONCONFIG %** file. If not, by default, onstat -c displays the contents of **%ONEDB_HOME%\etc\onconfig**.

onstat -C command: Print B-tree scanner information

Use the -C command to display information about the B-tree scanner subsystem and each B-tree scanner thread.

```
>>-onstat-- -C--+-prof--+--------------------------------------><
                +-hot---+
                +-part--+
                +-clean-+
                +-range-+
```

```
                 +-map---+
                 +-alice-+
                 '-all---'
```

The following options are available with the onstat -C command and can be combined:

prof

Prints the profile information for the system and each B-tree scanner thread. This is the default option.

hot

Prints the hot list index key in the order to be cleaned

part

Prints all partitions with index statistics

clean

Prints information about all the partitions that were cleaned or need to be cleaned

range

Prints the savings in pages processed by using index range scanning

map

Displays the current bitmaps for each index being cleaned by the alice cleaning method

alice

Displays the efficiency of the alice cleaning method option

all

Prints all onstat -C options

Example output using the prof option

Figure 58. onstat -C command output with the prof option

```
Btree Cleaner Info
BT scanner profile Information
==============================
Active Threads                              1
Global Commands                       2000000   Building hot list
Number of partition scans               11003
Main Block                         0xc000000003c9dc68
BTC Admin                          0xc0000000024bc208


BTS info             id   Prio    Partnum      Key     Cmd
0xc000000003c9dee8    0   High   0x00000000     0       40  Yield N
    Number of leaves pages scanned                        77
    Number of leaves with deleted items                    6
    Time spent cleaning (sec)                              0
    Number of index compresses                             0
    Number of deleted items                              113
    Number of index range scans                            0
    Number of index leaf scans                             0
    Number of index alice scans                            2
```

Output description using the prof option

Id

BTSCANNER ID

Prio

Current priority of BTSCANNER

Partnum

The partition number for the index this thread is currently working on

Cmd

Command this thread is processing currently

Example output using the hot option

Figure 59. onstat -C command output with the hot option

```

Btree Cleaner Info

Index Hot List
==============
    Current Item           5     List Created          15:29:47
    List Size              4     List expires in              0 sec
    Hit Threshold        500     Range Scan Threshold        -1


Partnum        Key            Hits
0x00100191       1              14 *
0x00A00022       1              13 *
0x00100191       2               8 *
0x00100150       2               7 *
```

Output description using the hot option

Partnum

The partition number for an index

Key

Index Key

Hits

The current value of the Hit counter

*****

Indicates that this partition has been cleaned during this hot list duration

Example output using the part option

Figure 60. onstat -C command output with the part option

```
Btree Cleaner Info


Index Statistics
================
 Partnum   Key     Positions   Compress      Split
0x00100002  1          146          0          0
0x00100004  1            4          0          0
0x00100004  2           13          0          0
0x00100005  1            1          0          0
0x00100005  2            0          0          0
0x00100006  1            1          0          0
0x00100006  2            0          0          0
0x00100007  2            1          0          0
0x00100008  2            1          0          0
0x0010000a  1            0          0          0
0x0010000e  3            1          0          0
0x00100011  1            1          0          0
0x00100013  2            2          0          0
```

Output description using the part option

Partnum

The partition number for an index

Key

Index Key

Positions

Number of times index has been read

Compress

Number of pages which have been compressed

Split

Number of splits that have occurred

C

Indicates partition is busy being cleaned

N

Index partition no longer eligible for cleaning

Example output using the clean option

Figure 61. onstat -C command output with the clean option

```
Btree Cleaner Info

Index Cleaned Statistics
=========================
 Partnum  Key      Dirty Hits  Clean Time  Pg Examined  Items Del  Pages/Sec
0x00100013  2              2           0            0          0       0.00
0x0010008b  3              1           0            0          0       0.00
0x001000c7  1              2           0            0          0       0.00
0x00100150  2              7           0            0          0       0.00
0x0010016f  2              2           0            0          0       0.00
0x00100191  1             14           0            0          0       0.00
0x00100191  2              8           0            0          0       0.00
0x00a00011  2              6           0            0          0       0.00
0x00a00013  1              0           0           24          0      24.00
0x00a00019  1              0           0          470        225     470.00
0x00a00022  1             13           0            0          0       0.00
0x00a00022  2              5           0            0          0       0.00
```

Output description using the clean option

Partnum

The partition number for an index

Key

Index Key

Dirty Hits

Number of times a dirty page has been scanned

Clean Time

Total time spent, in seconds

Pg Examined

Number of pages examined by btscanner thread

Items Del

Number of items removed form this index

Pages/Sec

Number of pages examined per second

C

Indicates partition is busy being cleaned

N

index partition is no longer eligible for cleaning

Example Output

Figure 62. onstat -C range

```
Btree Cleaner Info

Cleaning Range Statistics
=========================
 Partnum  Key             Low          High          Size        Saving
0x001001bc  2               36             69            96       65.6 %
0x001001be  1               16             20            48       91.7 %
0x001001cd  1                8             21            32       59.4 %
0x001001cd  2               24             25            32       96.9 %
```

Output Description

Partnum

The partition number

Key

Index Key

Low

Low boundary for range scan

High

High boundary for index scan

Size

Size of index in pages

Saving

Percentage of time saved versus a full scan

C

Indicates partition is busy being cleaned

N

Index partition is no longer eligible for cleaning

Example Output

Figure 63. onstat -C map

```
Btree Cleaner Info

ALICE Bitmap of Deleted Index Items
===================================
Partnum   Key        Map
0x00100013  2 0000: 80000000 00000000
0x0010008b  3 0000: 80000000 00000000
0x001000c7  1 0000: 80000000 00000000
0x00100150  2 0000: 80000000 00000000
0x0010016f  2 0000: 80000000 00000000
0x00100191  1 0000: 80000000 00000000
0x00100191  2 0000: 80000000 00000000
0x00a00011  2 0000: 80000000 00000000
0x00a00013  1 0000: 00000000 00000000
0x00a00019  1 0000: 00000000 00000000
0x00a00022  1 0000: 80000000 00000000
0x00a00022  2 0000: 80000000 00000000
```

Output Description

Partnum

The partition number

Key

Index Key

Map

Alice bitmap

Example Output

Figure 64. onstat -C alice

```
Btree Cleaner Info

ALICE Cleaning Statistics
=========================

System ALICE Info: Mode =    6, Eff =    30 %, Adj =     5

Partnum    Mode BM_Sz  Used_Pg  Examined  Dirty_Pg  # I/O  Found    Eff   Adj
0x00100013    6    64       97         0         0      0      0   0.0 %    0
0x0010008b    6    64        5         0         0      0      0   0.0 %    0
0x001000c7    6    64        2         0         0      0      0   0.0 %    0
0x00100150    6    64       91         0         0      0      0   0.0 %    0
0x0010016f    6    64       91         0         0      0      0   0.0 %    0
0x00100191    6    64       26         0         0      0      0   0.0 %    0
0x00100191    6    64       26         0         0      0      0   0.0 %    0
0x001001bc    0     0       91         0         0      0      0   0.0 %    0
0x001001cd    0     0       26         0         0      0      0   0.0 %    0
0x001001cd    0     0       26         0         0      0      0   0.0 %    0
0x00a00011    6    64       91         0         0      0      0   0.0 %    0
0x00a00013    6    64       25        24         3      3      1  33.3 %    1
0x00a00019    6    64      470       470         3      3      2  66.7 %    1
0x00a00022    6    64       26         0         0      0      0   0.0 %    0
0x00a00022    6    64       26         0         0      0      0   0.0 %    0
```

Output Description

Partnum

The partition number for an index

Mode

The alice mode for the current partition

BM_Sz

The size allocated for the bitmap

Used_Pg

The size of the index in pages (used)

Dirty_Pg

Number of dirty pages

 # I/O

Number of pages read

Found

Number of dirty pages found in reads

Eff

How efficient was the bitmap

Adj

Number of times the alice efficiency level for the partition was insufficient and was adjusted

onstat -d command: Print chunk information

Use the onstat -d command to show information about chunks in each storage space.

```
>-onstat-- -d--+--------+-------------------------------------><
                '-update-'
```

The update option updates shared memory to obtain accurate counts of free pages.

Using onstat -d with sbspaces

For information about using onstat -d to determine the size of sbspaces, user-data areas, and metadata areas, see Monitor sbspaces on page .

Using onstat -d with blobspaces

If you run the onstat -d command on a server that has blobspace chunks, the database server displays the following message:

```
NOTE: For BLOB chunks, the number of free pages shown is out of date.
 Run ‘onstat -d update' for current stats.
```

To obtain the current statistics for blobspace chunks, run the onstat -d update command. The onstat utility updates shared memory with an accurate count of free pages for each blobspace chunk. The database server shows the following message:

```
Waiting for server to update BLOB chunk statistics ...
```

Example output

Figure 65. onstat -d command output

```
Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
4484a028         1        0x10020001 1        1        2048     N  BA    informix rootdbs
45ed5b30         2        0x20001    2        1        2048     N  BA    informix space1
 2 active, 2047 maximum

Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
4484a268         1      1      0          100000     65632                 PO-B-- /work3/AB/rootchunk
45f1f028         2      2      0          5000       4947                  PO-B-- /work3/AB/chunk5
 2 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.


Expanded chunk capacity mode: always
```

Figure 66. onstat -d command output

```
BM Informix Dynamic Server Version 14.10.F       -- On-Line -- Up 00:01:27 -- 133540 Kbytes

Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
4484a028         1        0x10020001 1        1        2048     N  BA    informix rootdbs
45ed5b30         2        0x20001    2        1        2048     N  BA    informix space1
 2 active, 2047 maximum

Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
4484a268         1      1      0          100000     65632                 PO-B-- /work3/AB/rootchunk
45f1f028         2      2      0          5000       4947                  PO-B-- /work3/AB/chunk5
 2 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.


Expanded chunk capacity mode: always
```

Figure 67. onstat -d command output

```
BM Informix Dynamic Server Version 14.10.F       -- On-Line -- Up 00:01:27 -- 133540 Kbytes

Dbspaces
address          number   flags      fchunk   nchunks  pgsize   flags    owner    name
4484a028         1        0x10020001 1        1        2048     N  BAE   informix rootdbs
45ed5b30         2        0x20001    2        1        2048     N  BA    informix space1
 2 active, 2047 maximum

Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
4484a268         1      1      0          100000     65632                 PO-B-- /work3/AB/rootchunk
45f1f028         2      2      0          5000       4947                  PO-B-- /work3/AB/chunk5
 2 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.


Expanded chunk capacity mode: always
```

Output description for dbspaces

The first section of the output describes the storage spaces:

address

Is the address of the storage space in the shared-memory space table

number

Is the unique ID number of the storage space that is assigned at when it is created

flags

Uses hexadecimal values to describe each storage space. The individual flag values can be summed to show cumulative properties of the dbspace. The following table describes each hexadecimal value:

Table 154. Descriptions for each hexadecimal value

Flag Value	Description
0x0001	Mirror is allowed and dbspace is unmirrored.
0x0002	Mirror is allowed and dbspace is mirrored.
0x0004	The dbspace contains disabled mirror chunks.
0x0008	Newly mirrored
0x0010	Blobspace
0x0020	Blobspace on removable media
0x0040	Blobspace is on optical media

Table 154. Descriptions for each hexadecimal value (continued)

Flag Value	Description
0x0080	Blobspace is dropped.
0x0100	Blobspace is the optical STAGEBLOB
0x0200	Space is being recovered.
0x0400	Space is physically recovered.
0x0800	Logical log is being recovered.
0x1000	Table in dbspace is dropped.
0x2000	Temporary dbspace
0x4000	Blobspace is being backed up.
0x8000	Sbspace
0x10000	Physical or logical log changed.
0x20000	Dbspace or chunk tables changed.
0x040000	Blobspace contains large chunks.
0x080000	Chunk in this dbspace was renamed.
0x00100000	Temporary dbspace that is used by only by shared disk secondary server. It is one of the dbspaces listed in the SDS_TEMPDBS configuration parameter on the SD secondary server.
0x00200000	Temporary dbspace for the SD secondary server. Listed in the DBSPACETEMP configuration parameter on the shared disk secondary server.
0x00400000	The dbspace was externally backed up.
0x00800000	Dbspace is being defragmented.
0x01000000	Plogspace
0x10000000	The space is encrypted.

fchunk

The ID number of the first chunk

nchunks

The number of chunks in the storage space

pgsize

The size of the dbspace pages in bytes

flags

Uses the following letter codes to describe each storage space:

Position 1:

Flag	Description
M	Mirrored
N	Not mirrored

Position 2:

Flag	Description
X	Newly mirrored
P	Physically recovered, waiting for logical recovery
L	Being logically recovered
R	Being recovered
D	Down

Position 3:

Flag	Description
B	Blobspace
P	Plogspace
S	Sbspace
T	Temporary dbspace
U	Temporary sbspace
W	Temporary dbspace on primary server (This flag is shown on SD secondary servers only.)

Position 4:

Flag	Description
B	The dbspace can have large chunks that are greater than 2 GB.

Position 5:

Flag	Description
A	The dbspace is auto-expandable because the SP_AUTOEXPAND configuration parameter is enabled and the dbspace is configured with a create size or extend size that is not zero.

Position 6:

Flag	Description
E	The storage space is encrypted.

owner

The owner of the storage space

name

The name of the storage space

In the line immediately following the storage-space list, **active** refers to the current number of storage spaces in the database server instance, including the root dbspace and **maximum** refers to total allowable spaces for this database server instance.

Output description - Chunks

The second section of the onstat -d command output describes the chunks:

address

The address of the chunk

chk/dbs

The chunk number and the associated space number

offset

The offset into the file or raw device in base page size

size

The size of the chunk in terms of the page size of the dbspace to which it belongs.

free

The number of unallocated pages in the chunk in units of the page size of the associated dbspace. A value of `0` indicates that all the space in the chunk is allocated to tables, but does not indicate how much space is free inside the tables. For example, suppose you create a dbspace with one chunk of 200 MB and create one table with an extent size of 200 MB. The value of the **free** field is `0`, indicating that the chunk has no free space, however, the new empty table has 200 MB of free space.

For a blobspace, a tilde indicates an approximate number of unallocated blobpages.

For an sbspace, indicates the number of unallocated pages of user data space and total user data space.

Note: The "onstat -d" output adds footnotes for each of the down chunks that are empty, and a special foot note for the first chunk of a down space that is empty.

Example Output for onstat -d with down space (due to first chunk out of 4), where 2 of the 4 chunks are empty and can be dropped:

```
Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
44be2268         1      1      0          150000     83059
 PO-B-- /spaces/rootchunk
45c2b028         2      2      0          512        0
 PD-BE- /spaces/dbspace2_p_1
45c2c028         3      2      0          500        *
 PD-BE- /spaces/dbspace2_p_2
45c2d028         4      2      0          500        *
 PD-BE- /spaces/dbspace2_p_3
45c2e028         5      2      0          5000       0
 PD-BE- /spaces/dbspace2_p_4
5 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.

*  Down chunk is empty, and may be safely dropped.
```

Example Output for onstat -d when all chunks in down space are empty and can be dropped:

```
Chunks
address          chunk/dbs     offset     size       free       bpages     flags pathname
44be2268         1      1      0          150000     83059
 PO-B-- /spaces/rootchunk
45c2b028         2      2      0          512        **
 PD-BE- /spaces/dbspace2_p_1
45c2c028         3      2      0          500        *
 PD-BE- /spaces/dbspace2_p_2
45c2d028         4      2      0          500        *
 PD-BE- /spaces/dbspace2_p_3
45c2e028         5      2      0          5000       *
 PD-BE- /spaces/dbspace2_p_4
5 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.

*  Down chunk is empty, and may be safely dropped.
** Down space is empty, and may be safely dropped.
```

bpages

Is the size of the chunk in blobpages

Blobpages can be larger than disk pages; therefore, the **bpages** value can be less than the **size** value.

For an sbspace, is the size of the chunk in sbpages.

flags

Provides the chunk status information as follows:

Position 1:

Flag	Description
P	Primary
M	Mirror

Position 2:

Flag	Description
N	Renamed and either Down or Inconsistent
O	Online
D	Down
X	Newly mirrored
I	Inconsistent

Position 3:

Flag	Description
-	Dbspace
B	Blobspace
S	Sbspace

Position 4:

Flag	Description
B	The dbspace can have large chunks that are greater than 2 GB.

Position 5:

Flag	Description
E	Identifies the chunk as extendable
-	Identifies the chunk as not extendable

Position 6:

Flag	Description
-	The direct I/O or concurrent I/O option is not enabled for this cooked file chunk
C	On AIX®, the concurrent I/O option is enabled for this cooked file chunk
D	The direct I/O option is enabled for this cooked file chunk

pathname

The path name of the physical device

In the line immediately following the chunk list, **active** shows the number of active chunks (including the root chunk) and **maximum** shows the total number of chunks.

For information about page reads and page writes, run the onstat -D command.

onstat -D command: Print page-read and page-write information

Use the onstat -D command to display page-read and page-write information for the first 50 chunks in each space.

```
>>-onstat-- -D-------------------------------------------------><
```

Example output

Figure 68. onstat -D command output

```
Dbspaces
address  number   flags      fchunk   nchunks pgsize  flags    owner    name
a40d7d8  1        0x1        1        1       2048    N        informix rootdbs
 1 active, 2047 maximum

Chunks
address  chunk/dbs  offset     page Rd  page Wr  pathname
a40d928  1      1   0          0        0        /work/11.1/dbspaces/stardbs3
 1 active, 2047 maximum

Expanded chunk capacity mode: disabled
```

Output description

The output of onstat -D is almost identical to the output of onstat -d. The following columns are unique to onstat -D. For information on the other output columns see onstat -d command: Print chunk information on page 498.

page Rd

Is the number of pages read

page Wr

Is the number of pages written

onstat -f command: Print dbspace information affected by dataskip

Use the -f command to list the dbspaces that the dataskip feature currently affects.

```
>>-onstat-- -f-------------------------------------------------><
```

The -f option lists both the dbspaces that were set with the DATASKIP configuration parameter and the -f option of onspaces. When you execute onstat -f, the database server displays one of the following three outputs:

- Dataskip is OFF for all dbspaces.
- Dataskip is ON for all dbspaces.
- Dataskip is ON for the following dbspaces:

  ```
  dbspace1 dbspace2...
  ```

onstat -F command: Print counts

Use the onstat -F command to display a count for each type of write that flushes pages to disk.

Syntax:

onstat -F

Example output

Figure 70. onstat -F command output

```
Fg Writes     LRU Writes     Chunk Writes
0             330            7631

address  flusher  state  data  # LRU  Chunk  Wakeups  Idle Time
c7c8850  0        I      0     9      29     16116    16093.557
      states: Exit Idle Chunk Lru
```

Output description

You can interpret output from this option as follows:

Fg Writes

Is the number of times that a foreground write occurred

LRU *Writes*

Is the number of times that an LRU write occurred

Chunk Writes

Is the number of times that a chunk write occurred

address

Is the address of the user structure assigned to this page-cleaner thread

flusher

Is the page-cleaner number

state

Uses the following codes to indicate the current page-cleaner activity:

C

Chunk write

E

Exit

I

Cleaner is idle

L

LRU queue

The exit code indicates either that the database server is performing a shutdown or that a page cleaner did not return from its write in a specific amount of time. When an operation fails to complete within the allotted time, this situation is known as a time-out condition. The database server does not know what happened to the cleaner, so it is marked as `exit`. In either case, the cleaner thread eventually exits.

data

Provides additional information in concert with the **state** field

If **state** is `C`, **data** is the chunk number to which the page cleaner is writing buffers. If **state** is `L`, **data** is the LRU queue from which the page cleaner is writing. The **data** value is displayed as a decimal, followed by an equal sign, and repeated as a hexadecimal.

#LRU

Corresponds to the **onstat -g ath** thread ID output

Chunk

Number of chunks cleaned

Wakeups

Number of times the flusher thread was awoken

Idle Time

Time in seconds the flusher thread has been idle

onstat -g monitoring options

The options that you can use with onstat -g command are used for support and debugging only. You can include only one of these options in the onstat -g command.

onstat -g act command: Print active threads

Use the onstat -g act command to display information about the active threads.

Syntax:

onstat -gact

Following is sample output from the onstat -g act command. For a description of the output, see onstat -g ath command: Print information about all threads on page 512.

Example output

Figure 72. onstat -g act command output

```
Running threads:
tid  tcb        rstcb  prty  status     vp-class    name
2    b3132d8    0      1     running    *2adm       adminthd
40   c5384d0    0      1     running    *1cpu       tlitcppoll
```

onstat -g afr command: Print allocated memory fragments

Use the onstat -g afr command to display information about the allocated memory fragments for a specified session or shared-memory pool. Each session is allocated a pool of shared memory.

Syntax:

onstat -gafr { *pool_name* | *sessionid* | *pool_address* }

This command requires an additional argument to specify either a pool name, a session ID, or a pool address. Each session is allocated a memory pool with the same name as the session ID.

The *pool_name* is the name of the shared-memory pool. Run the onstat -g mem command to identify the pool name.

The *sessionid* is the session ID. Run the onstat -g ses command to identify the session ID.

The *pool_address* is the address of the shared-memory pool. Run the onstat -g mem command or the onstat -g ses command to identify the pool address.

Example output

Figure 74. onstat -g afr command output

```
Allocations for pool name global:
addr            size       memid        fileid   location
4b231000        3288       overhead     306      mtshpool.c:617
4b231cd8        72         mcbmsg       1637     rldmsg.c:92
4b231d20        160        mcbmsg       1637     rldmsg.c:92
4b231dc0        64         osenv        2909     osenv.c:1164
4b231e00        64         osenv        2909     osenv.c:1971
4b231e40        64         osenv        2909     osenv.c:1164
4b231e80        64         osenv        2909     osenv.c:1971
```

Output description

addr (hexadecimal)

Memory address of the pool fragment.

size (decimal)

Size, in bytes, of the pool fragment.

memid (string)

Memory ID of the pool fragment.

fileid (decimal)

Internal use only. Code file identifier for the allocation.

location (string)

Internal use only. Line number in the code for the allocation.

onstat -g all command: Print diagnostic information

Use the onstat -g all command to gather diagnostic information if advised to do so by HCL Support. For normal administrative purposes, use the onstat -g command with individual options.

Syntax:

```
onstat -gall
```

onstat -g arc command: Print archive status

Use the onstat -g arc command to display information about the last committed archive for each dbspace and also information about any current ongoing archives.

Syntax:

```
onstat -garc
```

Example output

Figure 77. onstat -g arc command output

```
Dbspaces - Ongoing archives
number   name           Q Size Q Len  buffer partnum    size     Current-page
1        rootdbs        100    3      100    0x1001c9   0        1:128
3        datadbs01      0      0
4        datadbs02      0      0

Dbspaces - Archive Status
name              number level date             log        log-position
rootdbs           1      0     07/30/2009.09:59 28         0x320018
datadbs01         3      0     07/30/2009.09:59 28         0x320018
datadbs02         4      0     07/30/2009.09:59 28         0x320018
```

Output description - Ongoing archives

This output section represents current information about the archives. If no archives are active in the system, this section is not displayed.

Column	Description
Number	The number of the dbspace
Name	The name of the dbspace
Q Size	The before-image queue list size. This information is primarily for the support.
Q Len	The before-image queue length. This information is primarily for the support.
Buffer	The number of pages used in the before-image buffer
Partnum	The partition number of the before-image bin
Size	The number of pages in the before-image bin
Current-page	The current page that is being archived

Note: The before-image bin is a temporary table created in a temporary dbspace, or in the root dbspace if you do not have any temporary dbspaces. If the before-image bin becomes too small, it can extend to additional partitions, in which case the output will display see multiple Partnum and Size fields for the same dbspace.

Output description - Archive status

This output section contains information about the last backup that has occurred for each dbspace.

Column	Description
Name	The name of the dbspace
Number	The dbspace number
Level	The archive level
Date	The date and time of the last archive
Log	The unique ID (UNIQID) of the checkpoint that was used to start the archive
Log-position	The log position (LOGPOS) of the checkpoint that was used to start the archive

onstat -g ath command: Print information about all threads

Use the onstat -g ath command to display information about all threads.

Syntax:

```
onstat  -gath
```

Example output

Figure 79. onstat -g ath command output

```
Threads:
 tid     tcb             rstcb       prty status            vp-class      name
 2       10bbf36a8        0           1    IO Idle            3lio         lio vp 0
 3       10bc12218        0           1    IO Idle            4pio         pio vp 0
 4       10bc31218        0           1    running            5aio         aio vp 0
 5       10bc50218        0           1    IO Idle            6msc         msc vp 0
 6       10bc7f218        0           1    running            7aio         aio vp 1
 7       10bc9e540        10b231028   1    sleeping secs: 1   1cpu         main_loop()
 8       10bc12548        0           1    running            1cpu         tlitcppoll
 9       10bc317f0        0           1    sleeping forever   1cpu         tlitcplst
 10      10bc50438        10b231780   1    IO Wait            1cpu         flush_sub(0)
 11      10bc7f740        0           1    IO Idle            8aio         aio vp 2
 12      10bc7fa00        0           1    IO Idle            9aio         aio vp 3
 13      10bd56218        0           1    IO Idle           10aio         aio vp 4
 14      10bd75218        0           1    IO Idle           11aio         aio vp 5
 15      10bd94548        10b231ed8   1    sleeping forever   1cpu         aslogflush
 16      10bc7fd00        10b232630   1    sleeping secs: 34  1cpu         btscanner 0
 32      10c738ad8        10b233c38   1    sleeping secs: 1   1cpu         onmode_mon
 50      10c0db710        10b232d88   1    IO Wait            1cpu         sqlexec
```

Output description

tid

Thread ID

tcb

Thread control block access

rstcb

RSAM thread control block access

prty

Thread priority

status

Thread status

vp-class

Virtual processor class

name

Thread name. For threads that are participating in parallel storage optimization operations, the name of the operation and the thread number.

- compress.*number* = The thread is compressing data
- repack.*number* = The thread is repacking data
- uncompress.*number* = The thread is uncompressing data
- update_ipa.*number* = The thread is removing outstanding in-place alter operations

onstat -g bth and -g BTH: Print blocked and waiting threads

Use the onstat -g bth command to display the dependencies between blocking and waiting threads. Use the onstat -g BTH command to display session and stack information for the blocking threads.

Syntax:

```
onstat -g { bth | BTH }
```

Example output for onstat -g bth

Figure 81. onstat -g bth command output

```
This command attempts to identify any blocking threads.

Highest level blocker(s)
 tid      name                 session
 48       sqlexec              26

Threads waiting on resources
 tid      name                 blocking resource              blocker
 49       sqlexec              MGM                            48
 13       readahead_0          Condition (ReadAhead)           -
 50       sqlexec              Lock (0x4411e578)              49
 51       sqlexec              Lock (0x4411e578)              49
 52       sqlexec              Lock (0x4411e578)              49
 53       sqlexec              Lock (0x4411e578)              49
 57       bf_priosweep()       Condition (bp_cond)             -
 58       scan_1.0             Condition (await_MC1)           -
 59       scan_1.0             Condition (await_MC1)           -

Run 'onstat -g BTH' for more info on blockers.
```

Ouput description for onstat -g bth

tid

Thread ID

name

Thread name

session

Session ID

blocking resource

Type of resource for which the listed thread is waiting

blocker

ID of the thread that is blocking the listed thread

Example output for onstat -g BTH

```
Stack for thread: 48 sqlexec
 base: 0x00000000461a3000
  len:   69632
   pc: 0x00000000017b32c3
  tos: 0x00000000461b2e30
state: ready
   vp: 1

0x00000000017b32c3 (oninit) yield_processor_svp
0x00000000017bca6c (oninit) mt_wait
0x00000000019d4e5c (oninit) net_buf_get
```

```
0x00000000019585bf (oninit) recvsocket
0x00000000019d1759 (oninit) tlRecv
0x00000000019ce62d (oninit) slSQIrecv
0x00000000019c43ed (oninit) pfRecv
0x00000000019b2580 (oninit) asfRecv
0x000000000193db2a (oninit) ASF_Call
0x0000000000c855dd (oninit) asf_recv
0x0000000000c8573c (oninit) _iread
0x0000000000c835cc (oninit) _igetint
0x0000000000c72a9e (oninit) sqmain
0x000000000194bb38 (oninit) listen_verify
0x000000000194ab8a (oninit) spawn_thread
0x0000000001817de3 (oninit) th_init_initgls
0x00000000017d3135 (oninit) startup
```

```
This command attempts to identify any blocking threads.

Highest level blocker(s)
 tid      name                 session
 48       sqlexec              26

session          effective                      #RSAM    total   used   dynamic
id      user     user      tty   pid   hostname threads  memory  memory explain
26      informix -         45    31041 mors     2        212992  186568 off

Program :
/work3/JC/VIEWS/jc_dct_phase2.view/.s/00055/80003fd351f804d3dbaccess

tid      name     rstcb            flags    curstk   status
48       sqlexec  448bc5e8         ---P---  4560     ready-
58       scan_1.0 448bb478         Y------  896      cond wait  await_MC1 -

Memory pools    count 2
name         class addr           totalsize  freesize   #allocfrag #freefrag
26           V     45fcc040       208896     25616      189        16
26*O0        V     462ad040       4096       808        1          1

name           free       used           name           free       used
overhead       0          6576           mtmisc         0          72
resident       0          72             scb            0          240
opentable      0          7608           filetable      0          1376
log            0          33072          temprec        0          17744
blob           0          856            keys           0          176
ralloc         0          55344          gentcb         0          2240
ostcb          0          2992           sqscb          0          21280
sql            0          11880          xchg_desc      0          1528
xchg_port      0          1144           xchg_packet    0          440
xchg_group     0          104            xchg_priv      0          336
hashfiletab    0          1144           osenv          0          2520
sqtcb          0          15872          fragman        0          1024
shmblklist     0          416            sqlj           0          72
rsam_seqscan   0          368

sqscb info
scb              sqscb          optofc   pdqpriority optcompind  directives
4499c1c0         461c1028       0        100         2           1

Sess       SQL            Current       Iso Lock       SQL  ISAM F.E.
```

```
Id          Stmt type      Database      Lvl Mode        ERR  ERR  Vers  Explain
26          SELECT         jc            CR  Not Wait    0    0    9.24  Off

Current statement name : unlcur

Current SQL statement (5) :
  select * from systables,syscolumns,sysfragments

Last parsed SQL statement :
  select * from systables,syscolumns,sysfragments
```

Ouput description for onstat -g BTH

tid

Thread ID

name

Thread name

session

Session ID

The session information section contains the same information that is output from the onstat -g ses command. See onstat -g ses command: Print session-related information on page 636.

The remainder of the information displays the stack information for the thread.

onstat -g buf command: Print buffer pool profile information

Use the onstat -g buf command to show profile information for each buffer pool.

Syntax:

```
onstat  -gbuf
```

Example output

Figure 83. onstat -g buf command output

```
Profile

Buffer pool page size: 2048
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
53056      118251     982617385  99.99   10640768   10915965   15968877    33.37

bufwrits_sinceckpt  bufwaits   ovbuff     flushes
17477               863        0          874

Fg Writes     LRU Writes    Avg. LRU Time   Chunk Writes
0             0             NaN             110767

Fast Cache Stats
gets       hits       %hits   puts
48314472   47220455   97.74   45757549
```

The output of the onstat -g buf command varies slightly depending on whether the BUFFERPOOL configuration parameter setting contains the memory field or the buffers field. The output for the memory setting is shown. The output for the buffers setting contains the **max extends** and **next buffers** fields instead of the **max memory** and **next memory** fields.

Figure 84. onstat -g buf output for the memory setting

```
Profile

Buffer pool page size: 2048
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
1190       1773       661359     99.82   16863      83049      185805     90.92
bufwrits_sinceckpt  bufwaits   ovbuff     flushes
11243               115        0          42

Fg Writes     LRU Writes    Avg. LRU Time Chunk Writes  Total Mem
0             0             nan           10883         32Mb

                                    cache
# extends  max memory  next memory  hit ratio  last
0          128Mb       32Mb         90         11:31:17

Bufferpool Segments
id segment      size     # buffs
0  0x449f0000   32Mb     13025

----------------------------------

Buffer pool page size: 8192
dskreads   pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
0          0          11         100.00  4          16         4          0.00
bufwrits_sinceckpt  bufwaits   ovbuff     flushes
0                   0          0          1

Fg Writes     LRU Writes    Avg. LRU Time Chunk Writes  Total Mem
0             0             nan           4             128Mb

                                    cache
# extends  max memory  next memory  hit ratio  last
0          1280Mb      128Mb        90         11:31:41

Bufferpool Segments
id segment      size     # buffs
0  0x4928e000   128Mb    14988

----------------------------------

Fast Cache Stats
gets       hits       %hits   puts
246854     244407     99.01   111147
```

Output description

Buffer pool page size

The number of bytes in a page in the buffer pool

dskreads

The number of disk read operations that are performed to bring pages into this buffer pool. Each read operation reads one or more pages.

pagreads

The number of pages that are read from disk to this buffer pool.

bufreads

The number of times a memory image for a page was read from this buffer pool.

%cached

The percentage of page reads for this buffer pool that were satisfied by a cached page image (rather than having to perform a disk read). Computed as `(bufreads - dskreads) / bufreads x 100`. Higher percentages indicate better caching performance.

dskwrits

The number of disk write operations that are performed to write changed pages from this buffer pool back to disk. Each write operation writes one or more pages.

pagwrits

The number of pages that are written to disk from this buffer pool.

bufwrits

The number of times a memory image of a page was written to in this buffer pool.

%cached

The percentage of page writes for this buffer pool that were satisfied by a cached page image (rather than having to perform a disk write). Computed as `(bufwrits - dskwrits) / bufwrits x 100`.

bufwrits_sinceckpt

The number of times a memory image of a page was written to in this buffer pool since the last checkpoint.

bufwaits

The number of times a thread had to wait for a lock on a buffer in this buffer pool. Higher numbers indicate more contention among multiple threads for mutually incompatible locks on the same pages.

ovbuff

The number of times a changed buffer from this buffer pool was written to disk specifically to create a free buffer to read another requested page. If the `ovbuff` value is high, the buffer pool might not be large enough to hold the working set of pages that are needed by applications. An insufficient buffer pool can lead to performance degradation.

flushes

The number of times the server flushed all dirty buffers at once in the buffer pool. Mass flushing can occur for various reasons, including as part of checkpoint processing or if the buffer pool is running out of clean buffers despite normal LRU cleaning activity.

Fg Writes

Number of changed buffers from this buffer pool that were written to disk by a non-I/O flusher thread that was accessing the buffer. This number is a superset of the value of the `ovbuff` field. In addition to the writes to service page faults that are counted in the `ovbuff` field, this value also includes foreground writes to maintain the consistency of database logs and reserved pages to ensure a correct recovery.

LRU Writes

The number of changed buffers from this buffer pool that were written to disk by an LRU cleaner thread. LRU cleaners are activated if the buffer pool exceeds the value that is specified in the lru_max_dirty field of the BUFFERPOOL configuration parameter or if foreground writes occur due to buffer pool overflows.

Avg. LRU Time

The average amount of time that is taken by an LRU cleaner thread to clean a single LRU chain.

Chunk Writes

The number of changed buffers that were written to disk by a chunk cleaning operation. Chunk cleaning writes out all changed buffers of a chunk that are in the buffer pool. Chunk cleaning is done to clean many buffers quickly, such as during checkpoint processing and fast recovery.

Total Mem

The size of the buffer pool.

 # extends

The number of times that the buffer pool was extended.

max memory (memory setting)

The target maximum size of the buffer pool. The actual size of the buffer pool can exceed this value, but not more than the size of one segment.

max extends (buffers setting)

The maximum number of times that the buffer pool can be extended. (This field is not shown in the example output.)

next memory (memory setting)

The size of the next extension of the buffer pool.

next buffers (buffers setting)

The number of buffers for the next extension of the buffer pool. (This field is not shown in the example output.)

cache hit ratio

The read cache hit ratio below which the buffer pool is extended.

last

The time of the last extension of the buffer pool.

id

The ID of the buffer pool segment.

segment

The internal address of the buffer pool segment.

size

The size of the buffer pool segment.

 # buffs

The number of buffers in the buffer pool segment.

Fast Cache Stats

Statistics for the fast cache, which is a type of cache that reduces the time that is needed for accessing the buffer pool.

gets

The number of times the server looked for a buffer in the fast cache.

hits

The number of times that the server found the buffer it was searching for in the fast cache.

%hits

The percentage of hits, which is `hits*100/gets`.

puts

The number of times that the server inserted buffers inserted into the fast cache.

onstat -g cac command: Print information about caches

Use the onstat -g cac command to see summary and detailed information about all caches or about a single cache.

Syntax:

onstat -gcac [{ **agg** | **aqt** | **am** *[access_method]* | **cast** | **dic** | **dsc** | **ed** | **lbacplcy** | **lbacusrc** | **poci** | **prc** | **prn** | **rr** | **ssc** | **ttype** | **typei** *[type_id]* | **typen** *[type_name]* }]

Use the onstat -g cac command without any options to see information about all caches.

Use the following options to see information about a specific cache:

agg

Prints information about the aggregate cache.

aqt

Prints information about the AQT dictionary cache. Prints the same output as the onstat -g aqt command. See #unique_635.

am

Prints information about the access method cache. To see information for a specific access method, include the access method name.

cast

Prints information about the cast cache.

dic

Prints information about the data dictionary cache. Prints the same output as the onstat -g dic command. See onstat -g dic command: Print table information on page 545.

dsc

Prints information about the data distribution cache. Prints the same output as the onstat -g dsc command. See onstat -g dsc command: Print distribution cache information on page 553.

ed

Prints information about the external directives cache.

lbacplcy

Prints information about the LBAC security policy information cache.

lbacusrc

Prints information about the LBAC credential memory cache.

opci

Prints information about the operator class instance cache.

prc

Prints information about the UDR cache. Prints the same output as the onstat -g prc command. See onstat -g prc command: Print sessions using UDR or SPL routines on page 606.

prn

Prints information about the procedure name cache.

rr

Prints information about the routine resolution cache.

ssc

Prints information about the SQL statement cache. Prints the same output as the onstat -g ssc command. See onstat -g ssc command: Print SQL statement occurrences on page 659.

ttype

Prints information about the secondary transient cache.

typei

Prints information about the extended type by ID cache. To see information for a specific extended type, include the extended type ID.

typen

Prints information about the extended type by name cache. To see information for a specific extended type, include the extended type name.

Example output

The output of most onstat -g cac commands contains similar format and information.

The following output is an example of the onstat -g cac lbacplcy command:

```
Security Policy Info Cache Entries:

list id   ref  drop hits       last_access          heap_ptr         item
------------------------------------------------------------------------------------

9    2    0    0    0          2020-05-12 10:36:34  65f1b8d0         test@informix: : secpolicyid 2

15   1    0    0    0          2020-05-12 10:36:34  65f1b4d0         test@informix: : secpolicyid 1



    Total number of entries  : 2
    Number of entries in use : 0
```

Output description

The output of most onstat -g cac commands contains the following fields:

Number of lists

Number of lists in the distribution cache

configuration parameter name

Number of entries that can be cached at one time

list

Distribution cache hash chain ID

id

The unique ID assigned to the cache entry

ref

Number of statements that reference a cache entry

drop

Whether this entry was dropped after it was added to the cache

hits

The number of times the cache entry is accessed

last_access

The time at which the cache entry was last accessed.

heap_ptr

Heap address that is used to store this entry

item name

The name of the item in the cache

Total number of entries

Number of entries in the cache

Number of entries in use

Number of entries that are being used

onstat -g ckp command: Print checkpoint history and configuration recommendations

Use the onstat -g ckp command to print checkpoint history and show configuration recommendations if a suboptimal configuration is detected.

Syntax:

```
onstat  -gckp
```

Example output

Figure 87. onstat -g ckp command output

```
Auto Checkpoints=On   RTO_SERVER_RESTART=60 seconds   Estimated recovery time 7 seconds

                                                             Critical Sections
         Clock                           Total  Flush  Block  #      Ckpt  Wait  Long  #Dirty
Interval Time     Trigger  LSN           Time   Time   Time   Waits  Time  Time  Time  Buffers
1        18:41:36 Startup  1:f8          0.0    0.0    0.0    0      0.0   0.0   0.0   4
2        18:41:49 Admin    1:11c12cc     0.3    0.2    0.0    1      0.0   0.0   0.0   2884
3        18:42:21 Llog     8:188         2.3    2.0    2.0    1      0.0   2.0   2.0   14438
4        18:42:44*User     10:19c018     0.0    0.0    0.0    1      0.0   0.0   0.0   39
5        18:46:21 RTO      13:188        54.8   54.2   0.0    30     0.6   0.4   0.6   68232

          Physical Log    Logical Log
  Dskflu  Total    Avg     Total    Avg
  /Sec    Pages    /Sec    Pages   /Sec
  4       3        0       1        0
  2884    1966     163     4549     379
  7388    318      10      65442    2181
  39      536      21      20412    816
  1259    210757   1033    150118   735

Max Plog    Max Llog    Max Dskflush    Avg Dskflush    Avg Dirty    Blocked
pages/sec   pages/sec   Time            pages/sec       pages/sec    Time
8796        6581        54              43975           2314          0
```

Output description

Auto Checkpoints

Indicates if the AUTO_CKPTS configuration parameter is on or off

RTO_SERVER_RESTART

Displays the RTO time in seconds. Zero (0) means that RTO is off.

Estimated recovery time ## seconds

Indicates the estimated recovery time if the data server stops responding. This value appears only if RTO_SERVER_RESTART is active.

Interval

Checkpoint interval ID

Clock Time

Clock time when checkpoint occurred.

Trigger

Event that triggered the checkpoint. An asterisk (*) indicates that the checkpoint that was requested was a transaction-blocking checkpoint.

Trigger name	Description
Admin	Administrator-related tasks. For example: • Create, drop, or rename a dbspace • Add or drop a chunk • Add or drop a log file • Change physical log size or location • After "shrink" operation on partition • Turn on or off mirroring
Backup	Back up related operations. For example: • Fake backup • Start of an archive • After the completion of a physical restore
CDR	ER subsystem is started for the first time, or is restarted after all of the replication participants were removed.
CKPTINTVL	When the checkpoint interval expires. The checkpoint interval is the value that is specified for the CKPTINTVL parameter in the `onconfig` file.
HA	High availability. For example:

Trigger name	Description
	• A new RSS or SDS node is added to a High Availability cluster • A secondary server is promoted to a primary server • The physical log file is low on a secondary server
HDR	High-Availability Data Replication. For example: • The mode of the server is changed • The start of the first transfer after HDR is set up • There is the potential for a physical log overflow on primary or secondary servers
IPL	Trigger checkpoint to reduce physical log usage on the secondary server. Index page logging can cause foreground writes and heavy physical log usage on secondary servers.
Lightscan	Before the look aside is turned off on partitions.
Llog	Running out of logical log resources.
LongTX	Long Transaction. If a long transaction was found but not stopped, a checkpoint is initiated to stop the transaction. During rollback, a checkpoint is initiated in the rollback phase if a checkpoint has not already happened after long transaction was aborted.
Misc	Miscellaneous events. For example: • A dbspace or chunk is being brought down because of I/O errors • During rollback when the addition of the chunk is being undone: for example, when removing the chunk.
Plog	Physical log has one of the following conditions:

Trigger name	Description
	• Physical log is 75% full • The amount of physical log used plus the number of dirty partitions is more than 90% of physical log size
Restore Pt	Restore Point. Checkpoints at the start and end of a restore point. The restore point is (used by conversion guard) CONVERSION_GUARD configuration parameter is enabled and a temporary directory is specified in the RESTORE_POINT_DIR configuration parameter.
Recovery	During a restore, at the start of a fast recovery.
Reorg	At the start of online index build.
RTO	Maintaining the Recovery Time Objective (RTO) policy. During normal operations, when the restart time after a crash might exceed the value that is set for the RTO_SERVER_RESTART configuration parameter.
Stamp Wrap	Checkpoint timestamp. If the new checkpoint timestamp appears to be before the last written checkpoint, then the timestamp is advanced out of interval between checkpoints. Another checkpoint is triggered.
Startup	At the startup of the database server.
Uncompress	Uncompress commands that are issued on a table or partition. This applies only for checkpoints on tables or databases that are not logged.
User	A checkpoint request is submitted by the user.

LSN

Logical log position where checkpoint is recorded

Total Time

Total checkpoint duration, in seconds, from request time to checkpoint completion

Flush Time

Time, in seconds, to flush buffer pools

Block Time

Time a transaction was blocked, in seconds, by a checkpoint that was triggered by a scarcity of some needed resource. For example, running out of physical log, or wrap-around of the logical log.

 # Waits

Number of transactions that are blocked waiting for checkpoint

Ckpt Time

Time, in seconds, for all transactions to recognize a requested checkpoint

Wait Time

Average time, in seconds, that transactions waited for checkpoint

Long Time

Longest amount of time, in seconds, a transaction waited for checkpoint

 # Dirty Buffers

Number of dirty buffers that are flushed to disk during checkpoint

Dskflu/sec

Number of buffers that are flushed per second

Physical Log Total Pages

Total number of pages that are physically logged during checkpoint interval

Physical Log Avg/Sec

Average rate of physical log activity during checkpoint interval

Logical Log Total Pages

Total number of pages that are logically logged during checkpoint interval

Logical Log Avg/Sec

Average rate of logical log activity during checkpoint interval

Max Plog pages/sec

Maximum rate of physical log activity during checkpoint interval

Max Llog pages/sec

Maximum rate of logical log activity during checkpoint interval

Max Dskflush Time

Maximum time, in seconds, to flush buffer pools to disk

Avg Dskflush pages/sec

Average rate buffer pools are flushed to disk

Avg Dirty pages/sec

Average rate of dirty pages between checkpoints

Blocked Time

Longest blocked time, in seconds, since the database server was last started

Performance advisory messages

If the HCL OneDB™ data server detects a configuration that is less than optimal, a performance advisory message with tuning recommendations appears below the checkpoint history. This performance advisory message also appears in the message log. Following are examples of performance advisory messages:

```
Physical log is too small for bufferpool size. System performance may be
less than optimal.
Increase physical log size to at least %ldKb

Physical log is too small for optimal performance.
Increase the physical log size to at least $ldKb.

Logical log space is too small for optimal performance.
Increase the total size of the logial log space to at least %ld Kb.

Transaction blocking has taken place. The physical log is too small.
Please increase the size of the physical log to %ldKb

Transaction blocking has taken place. The logical log space is too small.
Please increase the size of the logical log space to %ldKb
```

onstat -g cfg command: Print the current values of configuration parameters

Use the onstat -g cfg command to print a list of configuration parameters with their current values. You can use more command options to print more information about the configuration parameters.

Syntax:

onstat -gcfg [{ **full** | **diff** | **tunable** | **msg** }] [*config_parameter_name*]

This onstat -g cfg command has the following formats:

Command	Description
onstat -g cfg	Displays a list of configuration parameters and their current values.
onstat -g cfg *config_parameter_name*	Displays only the current value of the specified configuration parameter.
onstat -g cfg full	Displays all of the information about each configuration parameter, including the current value, the default value, the `onconfig` file value, and a description of the parameter.

Command	Description
onstat -g cfg full *config_parameter_name*	Displays all of the information about the specified parameter.
onstat -g cfg diff	Displays information about the configuration parameters with current values that are different from the permanent values that are in the `onconfig` file.
onstat -g cfg tunable	Displays the default, original, and current values for all tunable parameters. An asterisk indicates that you can tune a configuration parameter dynamically.
onstat -g cfg msg	Displays any messages, such as warnings or adjustments, that are associated with configuration parameters.

Example output

The following portion of sample output of the onstat -g cfg command shows that the value of the DEADLOCK_TIMEOUT configuration parameter was dynamically changed to 90 seconds after the database server started:

```
id   name                 type   units   rsvd  tunable
26   DEADLOCK_TIMEOUT     INT4   Seconds       *

     min/max : 0,2147483647
     default : 60
     onconfig:
     current : 90

     Description:
     Use the DEADLOCK_TIMEOUT configuration parameter to specify the
     maximum number of seconds that a database server thread can wait
     to acquire a lock.
```

```
ROOTNAME                 rootdbs
```

The following portion of sample output of the onstat -g cfg diff command shows the default, current, and `onconfig` file values of the TBLTBLFIRST and TBLTBLNEXT configuration parameters:

```
id   name                    type   units   rsvd  tunable
53   TBLTBLFIRST             INT4   KB            *

     default : 500
     onconfig: 0
     current : 250


id   name                    type   units   rsvd  tunable
54   TBLTBLNEXT              INT4   KB            *

     default : 100
     onconfig: 0
     current : 150
```

The following portion of sample output shows information for the MSGPATH configuration parameter. Here, there is no default value that is built into the configuration parameter and the `onconfig` file and current values are the same.

```
id   name                      type   units   rsvd  tunable
10   MSGPATH                   CHAR           *     *

     default :
     onconfig: /work2/JC/online.log
     current : /work2/JC/online.log
```

The following portion of sample output of the onstat -g cfg msg command shows messages that identify configuration parameters with changed values:

```
Configuration Parameters With Messages

name             message
TBLTBLFIRST      Parameter's user-configured value was adjusted.
TBLTBLNEXT       Parameter's user-configured value was adjusted.
BUFFERPOOL       Parameter's user-configured value was adjusted.
STACKSIZE        Parameter's user-configured value was adjusted.
VPCLASS          Parameter's user-configured value was adjusted.
```

Output description

name

Name of the configuration parameter

type

Data type for the value

units

Units in which the value is expressed

rsvd

Indicates (with an asterisk) that the configuration parameter and its value are stored on the configuration reserved page

If an asterisk is not present, the configuration parameter and its value are not stored on the configuration reserved page.

tunable

Indicates (with an asterisk) that the configuration parameter can be tuned dynamically, for example, with an onmode -wm or -wf command

If an asterisk is not present, the configuration parameter cannot be tuned dynamically.

min/max

Minimum and maximum values for the configuration parameter

default

Default value that is built into the server for the configuration parameter

onconfig

Value of the configuration parameter, if any, that is in the `onconfig.std` file

current

Current® value of the configuration parameter

A current value is different if it was modified dynamically, for example, with an onmode -wm command.

Description

Description of the configuration parameter

message

Message that identifies a changed configuration parameter value

onstat -g cluster command: Print high-availability cluster information

Use the onstat -g cluster command to display information about the servers in a high-availability cluster environment.

Syntax:

`onstat -gcluster [ { verbose } ]`

The onstat -g cluster command combines the functionality of onstat -g dri, onstat -g sds, and onstat -g rss. The output of the onstat -g cluster command differs slightly depending on whether the command is run on the primary server or on one of the secondary servers.

Example output (primary server)

Following is sample output from the onstat -g cluster command. The sample shows output when the command is run on the primary server.

Figure 90. onstat -g cluster command output (run on the primary server)

```
Primary Server:serv1
Current Log Page:16,476
Index page logging status: Enabled
Index page logging was enabled at: 2013/12/11 14:05:17


Server ACKed Log    Applied Log  Supports     Status
       (log, page)  (log, page)  Updates
serv2  16,476       16,476       Yes          SYNC(SDS),Connected,Active
serv3  16,476       16,476       Yes          ASYNC(HDR),Connected,On
serv4  16,476       16,476       Yes          ASYNC(RSS),Connected,Active
```

Output description (primary server)

Primary server

The name assigned to the primary server.

Current® log page

The log ID and page number of the current log page.

Index page logging status

Indicates whether index page logging is enabled or disabled.

Index page logging was enabled at

The date and time that index page logging was enabled.

Server

The name of the secondary server.

ACKed Log (log, page)

The log ID and page number of the last acknowledged log transmission.

Applied Log (log, page)

The log ID and page number of the last applied log transmission.

Supports Updates

Displays whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the UPDATABLE_SECONDARY configuration parameter).

Status

Displays the connection status of the secondary server

Example output (primary server, verbose output)

Figure 91. onstat -g cluster verbose command output (run on the primary server)

Following is sample output from the onstat -g cluster verbose command. The sample shows output when the command is run on the primary server with the verbose option.

```
Primary Server:serv1
 Current Log Page:16,479
 Index page logging status: Enabled
 Index page logging was enabled at: 2013/12/11 14:05:17



 -----------------------------------------------------------


 server name: serv3
 type: ASYNC (HDR)
 control block: 0x4b673018
 server status: On
 connection status: Connected
 Last log page sent (log id, page): 16,479
 Last log page acked (log id, page); 16,479
 Last log page applied (log id, page); 16,479
 Approximate log page backlog: 0
 SDS cycle not used
 Delayed Apply Not Used
 Stop Apply Not Used
 Time of last ack: 2013/12/11 14:09:12
 Supports Updates: Yes



 -----------------------------------------------------------


 server name: serv2
 type: SYNC (SDS)
 control block: 0x4c2de0b8
 server status: Active
 connection status: Connected
 Last log page sent (log id, page): 16,479
 Last log page acked (log id, page); 16,479
 Last log page applied (log id, page); 16,479
 Approximate log page backlog: 0
 SDS cycle current: 20 ACKed: 20
 Delayed Apply Not Used
 Stop Apply Not Used
 Time of last ack: 2013/12/11 14:09:13
 Supports Updates: Yes
```

Output description (primary server, verbose output)

Primary server

The name of the primary server

Current® log page

The log ID and page number of the current log page.

Index page logging status

Indicates whether index page logging is enabled or disabled.

Index page logging was enabled at

The date and time that index page logging was enabled.

Server name

The name of the secondary server.

type

Displays whether the secondary server is connected synchronously (SYNC) or asynchronously (ASYNC). Also displays the type of secondary server: HDR, SDS, or RSS.

control block

The in-memory address of the thread control block.

server status

Displays the current status of the secondary server.

connection status

Displays the current network connection status of the secondary server.

Last log page sent (log id, page)

The log ID and page number of the most recent log page sent by the primary server to the secondary server.

Last log page acked (log id, page)

The log ID and page number of the most recent log page the secondary server acknowledged.

Last log page applied (log id, page)

The log ID and page number of the most recent log page the secondary server applied.

Approximate log page backlog

Indicates the approximate number of log pages that have yet to be processed by the secondary server.

SDS cycle

Indicates the cycle number to which the primary server has advanced and which the shared disk secondary server has acknowledged. Used internally by the support to monitor coordination of the primary server with the secondary server.

Delayed Apply

Indicates whether the secondary server waits for a specified amount of time before applying logs (as specified by the DELAY_APPLY configuration parameter).

Stop Apply

Indicates whether the secondary server has stopped applying log files received from the primary server (as specified by the STOP_APPLY configuration parameter).

Time of last ack

The date and time of the last acknowledged log.

Supports Updates

Displays whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the UPDATABLE_SECONDARY configuration parameter).

Example output (secondary server)

Following is sample output from the onstat -g cluster command. The sample shows output when the command is run on a secondary server.

Figure 92. onstat -g cluster command output (run on the secondary server)

```
Primary Server:serv1
 Index page logging status: Enabled
 Index page logging was enabled at: 2010/01/11 14:05:17



 Server ACKed Log    Applied Log   Supports     Status
        (log, page)  (log, page)   Updates
 serv2  16,479       16,479        Yes          SYNC(SDS),Connected,Active
```

Output description (secondary server)

Primary server

The name of the primary server

Index page logging status

Indicates whether index page logging is enabled or disabled.

Index page logging was enabled at

The date and time that index page logging was enabled.

Server

The name of the secondary server.

ACKed Log (log, page)

The log ID and page number of the last acknowledged log.

Applied Log (log, page)

The log ID and page number of the last applied log transmission.

Supports Updates

Displays whether client applications can perform update, insert, and delete operations on the secondary server (as specified by the UPDATABLE_SECONDARY configuration parameter).

Status

Displays the connection status of the secondary server.

onstat -g cmsm command: Print Connection Manager information

Use the onstat -g cmsm command to display information about a specific Connection Manager, or all of the Connection Managers that are attached to the database server the command is run on.

Syntax:

onstat -gcmsm *[connection_manager_name]*

Usage

onstat -g cmsm displays information about connection units the Connection Manager connects to, the number of connections each Connection Manager service-level-agreement (SLA) has processed, SLA definitions, failover-order rules, failover arbitration, and primary server status.

Use *connection_manager_name* to display information for a specific Connection Manager instance. If *connection_manager_name* is not specified, onstat -g cmsm displays information about all Connection Manager instances that are connected to the database server.

Example output 1: Output for a specific Connection Manager

In the following example, onstat -g cmsm connection_manager_1 is run on the primary server of **my_cluster_1**.

```
Unified Connection Manager: connection_manager_1                Hostname: my_host_1

CLUSTER         my_cluster_1      LOCAL
   SLA          Connections    Service/Protocol   Rule
   oltp_1                35     19910/onsoctcp    DBSERVERS=primary
   report_1              33     19810/onsoctcp    DBSERVERS=(HDR,SDS,RSS)

   Failover Arbitrator: Active Arbitrator, Primary is up
   ORDER=SDS,HDR,RSS PRIORITY=1
```

The command displays output for **connection_manager_1**. **connection_manager_1** manages a CLUSTER connection unit, and is the active failover arbiter.

Example output 2: Output for a high-availability cluster

In the following example, onstat -g cmsm is run on the primary server of **my_cluster_2**.

```
Unified Connection Manager: connection_manager_2                Hostname: my_host_2

CLUSTER         my_cluster_2      LOCAL
   SLA          Connections    Service/Protocol   Rule
   sla_1               1535     19910/onsoctcp    DBSERVERS=primary
   sla_2               2133     19810/onsoctcp    DBSERVERS=(HDR,SDS,RSS)

        Failover Arbitrator: Active Arbitrator, Primary is up
        ORDER=SDS,HDR,RSS PRIORITY=1

CLUSTER         my_cluster_3
```

```
   SLA          Connections   Service/Protocol   Rule
   sla_3                730     19930/onsoctcp   DBSERVERS=primary
   sla_4                901     19830/onsoctcp   DBSERVERS=(HDR,SDS,RSS)

   Failover Arbitrator: Active Arbitrator, Primary is up
   ORDER=SDS,HDR,RSS PRIORITY=1

Unified Connection Manager: connection_manager_3                Hostname: my_host_3

CLUSTER         my_cluster_2      LOCAL
   SLA          Connections   Service/Protocol   Rule
   sla_5                614     19920/onsoctcp   DBSERVERS=primary
   sla_6                483     19820/onsoctcp   DBSERVERS=(HDR,SDS,RSS)

   Failover Arbitrator: Failover is enabled
   ORDER=SDS,HDR,RSS PRIORITY=2

CLUSTER         my_cluster_3
   SLA          Connections   Service/Protocol   Rule
   sla_7                678     19940/onsoctcp   DBSERVERS=primary
   sla_8                270     19840/onsoctcp   DBSERVERS=(HDR,SDS,RSS)

        Failover Arbitrator: Failover is enabled
        ORDER=SDS,HDR,RSS PRIORITY=2
```

The command displays output for the two Connection Managers that connect to the primary server of the cluster. **connection_manager_2** and **connection_manager_3** are installed on separate hosts, and together they manage two CLUSTER connection units. **connection_manager_2** is the active failover arbiter for both CLUSTER connection units.

Example 3: Output for a replicate set

In the following example, onstat -g cmsm is run on a replicate server in **my_replicate_set_1**.

```
Unified Connection Manager: connection_manager_4                Hostname: my_host_4

REPLSET         my_replicate_set_1
   SLA          Connections   Service/Protocol   Rule
   sla_1                160     19810/onsoctcp   DBSERVERS=ANY

Unified Connection Manager: connection_manager_5                Hostname: my_host_5

REPLSET         my_replicate_set_1
   SLA          Connections   Service/Protocol   Rule
   sla_2                240     19820/onsoctcp   DBSERVERS=ANY
```

The command displays output for the two Connection Managers that connect to the replicate server. **connection_manager_4** and **connection_manager_5** are installed on separate hosts, and together they manage the replication servers.

Example 4: Output for a grid

In the following example, onstat -g cmsm is run on a node of **my_grid_1**.

```
Unified Connection Manager: connection_manager_6                Hostname: my_host_6

GRID    my_grid_1
   SLA          Connections   Service/Protocol   Rule
   sla_1                456     19830/onsoctcp   DBSERVERS=(group_name_1,group_name_2)
 POLICY=FAILURE

Unified Connection Manager: connection_manager_7                Hostname: my_host_7

GRID    my_grid_1
   SLA          Connections   Service/Protocol   Rule
   sla_2                785     19840/onsoctcp   DBSERVERS=(group_name_1,group_name_2)
 POLICY=FAILURE
```

The command displays output for the two Connection Managers that connect to the grid. The command displays output for the two Connection Managers that connect to the node. **connection_manager_6** and **connection_manager_7** are installed on separate hosts, and together they manage the grid.

Example 5: Output for a server set

In the following example, onstat -g cmsm is run on a stand-alone server in the server set.

```
Unified Connection Manager: connection_manager_8                      Hostname: my_host_8

SERVERSET       server_1,server_2
   SLA          Connections   Service/Protocol   Rule
   sla_1                 63     19810/onsoctcp   DBSERVERS=(server_1,server_2) POLICY=ROUNDROBIN

Unified Connection Manager: connection_manager_9                      Hostname: my_host_9

SERVERSET       server_1,server_2
   SLA          Connections   Service/Protocol   Rule
   sla_2                 63     19810/onsoctcp   DBSERVERS=(server_1,server_2) POLICY=ROUNDROBIN
```

The command displays output for the two Connection Managers that connect to the server set. **connection_manager_8** and **connection_manager_9** are installed on separate hosts, and together they manage the server set.

Output description

The output of the onstat -g cmsm command contains sections for each Connection Manager. Each section displays the Connection Manager instance name and host name, followed by subsections that contain information on each connection unit the Connection Manager connects to.

Unified Connection Manager

The name of the Connection Manager instance.

Hostname

The name of the Connection Manager's host.

SLA

The names of service level agreements, as defined in the Connection Manager's configuration file.

Connections

The numbers of connections each SLA processed since the Connection Manager started.

Service/Protocol

The port number or service name that is associated with the SLA, followed by the connection protocol type.

Rule

The SLA definition.

Failover Arbitrator:

Specifies whether the Connection Manager is the active failover arbiter, if the primary server is active, and if failover is enabled. Displays only for CLUSTER connection units.

ORDER

Specifies the failover order for a cluster. Displays only for CLUSTER connection units.

PRIORITY

Specifies the priority of the connection between the Connection Manager and the primary server of a cluster. Displays only for CLUSTER connection units.

onstat -g con command: Print condition and thread information

Use the onstat -g con command to display information about conditions and the threads that are waiting for the conditions.

Syntax:

```
onstat -gcon
```

Example output

Figure 95. onstat -g con command output

```
Conditions with waiters:
cid      addr            name             waiter   waittime
271      c63d930         netnorm          1511     6550
```

Output description

cid

Condition identifier

addr

Condition control block address

name

Name of condition the thread is waiting on

waiter

ID of thread waiting on condition

waittime

Time, in seconds, thread has been waiting on this condition

onstat -g cpu: Print runtime statistics

Use the onstat -g cpu command to display information about runtime statistics for each thread that is running in the server.

Syntax:

```
onstat  -gcpu
```

Example output

Figure 97. onstat -g cpu command output

```
onstat -g cpu

Thread CPU Info:
tid     name              vp       Last Run           CPU Time     #scheds     status
2       lio vp 0          3lio*    07/18 08:35:35       0.0000           1     IO Idle
3       pio vp 0          4pio*    07/18 08:35:36       0.0102           2     IO Idle
4       aio vp 0          5aio*    07/18 08:35:47       0.6876          68     IO Idle
5       msc vp 0          6msc*    07/18 11:47:24       0.0935          14     IO Idle
6       main_loop()       1cpu*    07/18 15:02:43       2.9365       23350     sleeping secs: 1
7       soctcppoll        7soc*    07/18 08:35:40       0.1150           1     running
8       soctcpio          8soc*    07/18 08:35:40       0.0037           1     running
9       soctcplst         1cpu*    07/18 11:47:24       0.1106          10     sleeping forever
10      soctcplst         1cpu*    07/18 08:35:40       0.0103           6     sleeping forever
11      flush_sub(0)      1cpu*    07/18 15:02:43       0.0403       23252     sleeping secs: 1
12      flush_sub(1)      1cpu*    07/18 15:02:43       0.0423       23169     sleeping secs: 1
13      flush_sub(2)      1cpu*    07/18 15:02:43       0.0470       23169     sleeping secs: 1
14      flush_sub(3)      1cpu*    07/18 15:02:43       0.0407       23169     sleeping secs: 1
15      flush_sub(4)      1cpu*    07/18 15:02:43       0.0307       23169     sleeping secs: 1
16      flush_sub(5)      1cpu*    07/18 15:02:43       0.0323       23169     sleeping secs: 1
17      flush_sub(6)      1cpu*    07/18 15:02:43       0.0299       23169     sleeping secs: 1
18      flush_sub(7)      1cpu*    07/18 15:02:43       0.0314       23169     sleeping secs: 1
19      kaio              1cpu*    07/18 14:56:42       1.4560     2375587     IO Idle
20      aslogflush        1cpu*    07/18 15:02:43       0.0657       23166     sleeping secs: 1
21      btscanner_0       1cpu*    07/18 15:00:53       0.0484         784     sleeping secs: 61
37      onmode_mon        1cpu*    07/18 15:02:43       0.3467       23165     sleeping secs: 1
43      dbScheduler       1cpu*    07/18 14:58:14       1.6613         320     sleeping secs: 31
44      dbWorker1         1cpu*    07/18 13:48:10       0.4264         399     sleeping forever
45      dbWorker2         1cpu*    07/18 14:48:11       1.9346        2936     sleeping forever
94      bf_priosweep()    1cpu*    07/18 15:01:42       0.0431          77     cond wait   bp_cond
```

Output description

tid

The ID of the thread

name

The name of the thread

vp

The ID of the virtual processor in which the thread is running

Last Run

The timestamp when the thread last ran

CPU Time

The time taken until now by the thread

#scheds

The number of times the thread was scheduled to run

status

The status of the thread. Possible status values are:

- cond wait
- IO Idle
- join wait
- mutex wait
- ready
- sleeping
- terminated
- running
- yield

onstat -g dbc command: Print dbScheduler and dbWorker thread statistics

Use the onstat -g dbc command to display statistics about the Scheduler tasks that are currently running, which are handled by dbWorker threads, or scheduled to be run, which are handled by the dbScheduler thread.

Syntax:

```
onstat  -gdbc
```

Example output

Figure 99. onstat -g dbc command output

```
Worker Thread(0)    46fa6f10
====================================
Task:               47430c18
Task Name:          mon_config_startup
Task ID:            3
Task Type:          STARTUP SENSOR
Last Error
    Number              -310
    Message             Table (informix.mon_onconfig) already exists in database.
    Time                09/11/2007 11:41
    Task Name           mon_config_startup

Task Execution:     onconfig_save_diffs

WORKER PROFILE
    Total Jobs Executed            10
    Sensors Executed                8
    Tasks Executed                  2
    Purge Requests                  8
    Rows Purged                     0
```

```
Worker Thread(1)    46fa6f80
====================================
Task:               4729fc18
Task Name:          mon_sysenv
Task ID:            4
Task Type:          STARTUP SENSOR
Task Execution:     insert into mon_sysenv select 1, env_name, env_value FROM
                    sysmaster:sysenv


WORKER PROFILE
    Total Jobs Executed             3
    Sensors Executed                2
    Tasks Executed                  1
    Purge Requests                  2
    Rows Purged                     0


Scheduler Thread    46fa6f80
====================================
Run Queue
    Empty
Run Queue Size       0
Next Task            7
Next Task Waittime    57
```

Output description

Worker Thread

Address of the worker thread in shared memory

Task

Name of the last executed task

Task ID

The task ID from the **tk_id** column in the **sysadmin:ph_task** table for this task

Task Type

Type of the task

Last Error

Error number, error message, time (in seconds), and task name from the last error the dbWorker thread encountered. It could be from the previously executed task or from a task executed days ago.

Task Execution

SQL statement or SPL procedure or routine executed as part of the task

WORKER PROFILE

The dbWorker thread profile data shows the total jobs executed, number of sensors executed, number of tasks executed, number of purge requests, and the number of rows purged from the result tables for all sensors executed by this dbWorker thread.

Scheduler Thread

Address of the scheduler thread in shared memory

Run Queue

The task ID for the next scheduled task. If no task is scheduled, the value is `Empty`.

Run Queue Size

The number of tasks that are waiting to be executed by the dbWorker thread

Next Task

The task ID of the next task that will be scheduled to be executed

Next Task Waittime

The number of seconds before the `Next Task` will be scheduled for execution

onstat -g defragment command: Print defragment partition extents

Use the onstat -g defragment command to display information about the active requests to defragment partition extents.

Syntax:

```
onstat -gdefragment
```

Example output

Figure 101. onstat -g defragment command output

```
Defrag info
id  table name                     tid  dbsnum  partnum  status               substatus  errnum
15  stores_demo:informix.stdtab2   49   2       2097155  SEARCHING_FOR_EXTENT 0          0
```

Note: This command displays information about defragment requests that are active. If there are no active defragment requests, only the column headings are returned.

Output description

id

The ID of the defragment request.

table name

The fully-qualified name of the table that is being defragmented.

tid

The thread ID.

dbsnum

The dbspace number that is being defragmented.

partnum

The partition number that is being defragmented.

status

- SEARCHING_FOR_EXTENT
- MERGING_EXTENTS
- DEFRAG_COMPLETED
- DEFRAG_FAILED

substatus

The detailed status number, if any.

errnum

The last error number returned from the defragmentation request.

onstat -g dic command: Print table information

Use the onstat -g dic command to display a line of information about each table that is cached in the shared-memory dictionary. If you specify a table name, this command prints internal SQL information about that particular table.

Syntax:

```
onstat -gdic
```

Example output

Figure 103. onstat -g dic command output

```
Dictionary Cache:  Number of lists: 31, Maximum list size: 10
list#  size  refcnt  dirty?  heapptr        table name
--------------------------------------------------------
   1     3      1      no     14b5d890   wbe@oninit_shm:informix.t0010url
                1      no     14cbb820   wbe@oninit_shm:informix.t9051themeval
                0      no     14b63c20   wbe@oninit_shm:informix.t0060hits

   2     2      0      no     14b97420   wbe@oninit_shm:informix.t0120import
                1      no     14b6c820   wbe@oninit_shm:informix.t9110domain

   3     3      0      no     14bce020   wbe@oninit_shm:informix.t0150url
                0      no     14d3d820   contact@oninit_shm:informix.wbtags
                0      no     14c87420   wbe@oninit_shm:informix.wbtags

   4     1      0      no     14b7a420   drug@oninit_shm:abcdef.product      ..
Total number of dictionary entries: 36
```

Output description

list#

Data dictionary hash chain ID

size

Number of entries in this hash

refcnt

Number of SQL statements currently referencing one of the cache entries.

dirty?

Whether the entry has been modified since last written to disk.

heapptr

Address for the heap used to store this table

table name

Name of table in cache

onstat -g dis command: Print database server information

Use the onstat -g dis command to display a list of database servers, the status of each server, and information about each server, including the location of the **ONEDB_HOME** directory, **sqlhosts** file, and ONCONFIG file. You can use this command in any database server mode, including offline.

Syntax:

```
onstat -gdis
```

Example output

Figure 105. onstat -g dis command output

```
There are 2 servers found
Server        : ol_tuxedo
Server Number : 53
Server Type   : IDS
Server Status : Up
Server Version: IBM Informix Version 11.50.UC1
Shared Memory : 0xa000000
ONEDB_HOME   : /local1/engines/ol_tuxedo/dist
ONCONFIG      : /local1/engines/ol_tuxedo/dist/etc/onconfig.ol_tuxedo
SQLHOSTS      : /local1/engines/ol_tuxedo/dist/etc/sqlhosts
Host          : avocet

Server        : ol_9next
Server Number : 0
Server Type   : IDS
Server Status : Down
Server Version:
Shared Memory : 0
ONEDB_HOME   : /local1/engines/ol_9next/dist
ONCONFIG      :
SQLHOSTS      :
Host          :
```

Output description

Server

Server name

Server Number

Number of the server.

Server Type

Type of server

Server Status

Up means that the server is online, Down means that the server is offline

Server Version

Version of the server

Shared Memory

Location of the shared memory address

ONEDB_HOME

Location of the **$ONEDB_HOME/** directory on UNIX™ and in the **%ONEDB_HOME%** directory on Windows™.

ONCONFIG

Location of the ONCONFIG file

SQLHOSTS

Location of the **sqlhosts** file

Host

Host name of the server

onstat -g dll command: Print dynamic link library file list

Use the onstat -g dll command to display a list of and the status of dynamic link library (DLL) files that were loaded.

Syntax:

```
onstat  -gdll
```

Example output

The output displays the names of the library files only one time each process group. The flags indicate if the library was loaded when the server was started.

Figure 107. onstat -g dll command output

```
addr          slot  vp baseaddr        flags   filename
0x4af55310    15     1  0x2a985e3000    PM      /finance/jeffzhang/mylib.udr
0x4b6f2310           2  0x2a985e3000
0x4b71b310           3  0x2a985e3000
0x4c09f310    16     1  0x2a985e3000    M       /deptxyz/udrs/geodetic.bld
0x4c0c0310           2  0x2a985e3000
0x4c0f1310           3  0x2a985e3000
0x4c112310    17     1  0x7a138e9000            /home/informix/extend/blade.so
0x4c133310           2  0x3a421e1000
0x4c133310           3  0x3a421e1000
```

Output description

addr

Address of the DLL file

slot

Slot number entry in the library table

vp

ID of the virtual processor

baseaddr

Base address of the shared library

flags

- M indicates that the thread calling the UDR can migrate from one CPU virtual processor to another CPU virtual processor.
- P indicates that the shared library was loaded when the database server was started.

filename

Name of the DLL file

onstat -g dmp command: Print raw memory

Use the onstat -g dmp command to display information about raw memory at a given address for a number of given bytes.

Syntax:

onstat -gdmp*addresslength*

Each address and length must be within the allocated memory shown from **onstat -g seg** output. The address specified can be in decimal or hexadecimal format. Hexadecimal addresses must begin with 0x. You can specify the address in decimal, but doing so requires converting the memory shown from **onstat -g seg** to decimal before using it as a command line argument.

Example output

Figure 109. onstat -g dmp command output

```
%onstat -g dmp 0x700000011a19d48 100

address           bytes in mem
0700000011a19d48: 07000000 118e0fa8 07000000 11942b40 ........ ......+@
0700000011a19d58: 07000000 10137120 00000000 00000000 ......q ........
0700000011a19d68: 00000000 00000000 00000000 00000000 ....... ........
0700000011a19d78: 07000000 11a19d48 07000000 11a19d48 .......H .......H
0700000011a19d88: 00000000 00000000 00000000 00000000 ........ ........
0700000011a19d98 *
0700000011a19da8: 00000000 ....
```

Output description

address

Memory address of the raw memory.

bytes in mem

Hexadecimal and ASCII representations of the memory contents.

Output from the command is divided into three columns: memory address, hexadecimal values for the bytes in memory, and the ASCII representation of the bytes in memory. The bytes in memory (middle) section displays the first 16 bytes of memory starting at the address specified on the command line. The third column shows the ASCII representation of the hexadecimal data. Periods are displayed for all hexadecimal values that do not have an ASCII character equivalent. ASCII values are shown in order to make searching for plain text easier.

In the example output shown, the fifth line of data displays zeros and the sixth line contains an asterisk. The asterisk indicates an unknown number of repetitions of the previous line, which means that there is no more data after the fourth line.

onstat -g dri command: Print high-availability data replication information

Use the onstat -g dri command, either alone or with the ckpt or que options, to print information about high-availability data replication statistics on the current server.

Use the onstat -g dri command to print information about HDR server states and HDR-related configuration parameters.

Syntax:

onstat -gdri [{ ckpt | que }]

Example output and output description for onstat -g dri

Figure 111. onstat -g dri command output

```
Data Replication at 0x4d676028:
  Type           State        Paired server        Last DR CKPT (id/pg)    Supports Proxy Writes
  primary        on           my_server                     4 / 5           NA

  DRINTERVAL   5
  DRTIMEOUT    30
  DRAUTO       3
  DRLOSTFOUND  /etc/dr.lostfound
  DRIDXAUTO    0
  ENCRYPT_HDR  0
  Backlog      0
  Last Send    2013/12/11 16:39:48
  Last Receive 2013/12/11 16:39:48
  Last Ping    2013/12/11 16:39:44
  Last log page applied(log id,page): 4,6
```

Type

Current® type of server: primary, secondary, or standard

State

on or off

Paired server

Name of the primary or secondary server that this server is paired with

Last DR CKPT

Last checkpoint ID and page

Supports Proxy Writes

Displays whether the server is configured to allow secondary server updates. **Y** = supports secondary server updates, **N** = does not support secondary server updates.

DRINTERVAL

The value of the configuration parameter in the `onconfig` file.

DRTIMEOUT

The value of the configuration parameter in the `onconfig` file.

DRAUTO

The value of the configuration parameter in the `onconfig` file.

DRLOSTFOUND

The value of the configuration parameter in the `onconfig` file.

DRIDXAUTO

The value of the configuration parameter in the `onconfig` file.

ENCRYPT_HDR

The value of the configuration parameter in the `onconfig` file.

Backlog

Number of log pages in the HDR data replication buffer that are not yet sent to the HDR secondary server

Last Send

The time that the last message was sent to the peer node

Last Receive

The time that the last message was received from the peer node

Last Ping

The time of the last ping

Last log page applied(log id,page)

The log ID and page number of the last applied log

Example output and output description for onstat -g dri ckpt

Use the onstat -g dri ckpt command to print information about nonblocking checkpoints in HDR servers.

Figure 112. onstat -g dri ckpt command output

```
Data Replication:
  Type           State        Paired server        Last DR CKPT (id/pg)   Supports Proxy Writes
  primary        on           BB_1                        554 / 558       Y

  DRINTERVAL   30
  DRTIMEOUT    30
  DRAUTO       0
  DRLOSTFOUND  /vobs/tristarm/sqldist/etc/dr.lostfound
  DRIDXAUTO    0
  ENCRYPT_HDR  0

DR Checkpoint processing:
  Save State             N
  Pages Saved            0
  Save Area              none
  Received log id, page   17,68
  Saved log id, page     0,0
  Drain log id, page     0,0
  Processed log id, page  17,68
  Pending checkpoints    0
```

Save State

B (buffering) when the server is adding logs to the staging area

D (draining) when the server is removing logs from the staging area

N (normal) when the server is operating normally, meaning that no logs are saved

Pages Saved

Displays the number of log pages saved in the staging area that have yet to be applied.

Save Area

Displays the location of the staged log files.

Received log id, page

Displays the last log ID and page that were received from the primary server.

Processed log id, page

Displays the last log ID and page that are queued to the recovery pipeline.

Saved log id, page

Displays the last log ID and page that was stored in the staging area (if stage state is either **B** or **D**).

Drain log id, page

Displays the last log ID and page that were removed from the staging area.

Pending checkpoints

Displays the number of checkpoints that are staged but not yet applied.

Pending ckpt log id, page

Displays the position of any pending checkpoint records.

Example output and output description for onstat -g dri que

Use the onstat -g dri que command to print information that is related to nearly synchronous HDR replication.

Figure 113. onstat -g dri que command output

```
Pending Msg to Send 1
ACK QUEUE  5199:1256fff
thread 0x893de6c8 (85) 5199:1258018
thread 0x893a16b8 (83) 5199:1258048
thread 0x89229968 (72) 5199:1258078
thread 0x89381508 (82) 5199:12580a8
thread 0x87e81658 (69) 5199:12580d8
thread 0x89215968 (71) 5199:1259018
thread 0x89336bc8 (80) 5199:1259048
thread 0x89370018 (81) 5199:12590f8
thread 0x892eb018 (77) 5199:125a018
thread 0x89308018 (78) 5199:125b018
thread 0x89290138 (75) 5199:125b048
thread 0x893c1658 (84) 5199:125c018
thread 0x891fe8e8 (70) 5199:125c048
thread 0x89325018 (79) 5199:125d018
thread 0x893ff738 (86) 5199:125d048
thread 0x894207a8 (87) 5199:125d078

Applied QUEUE  5199:1251018
-----------------------------------------
```

Pending message to send

The number of unprocessed data replication buffers queued to the **drprsend** thread.

ACK QUEUE

The log unique value, the page number, and the value 0xfff for the most recently paged log.

thread

The pointer to the thread-control block (TCB), the thread id in parentheses, and the log sequence number (LSN) of the commit that was performed by that thread

Applied QUEUE

The LSNs of commits that are waiting for acknowledgement of being received on the HDR secondary.

onstat -g dsc command: Print distribution cache information

Use the onstat -g dsc command to display information about the distribution cache.

Syntax

```
onstat  -gdsc
```

Example output

Figure 115. onstat -g dsc command output

```
Data Distribution Cache:
    Number of lists         : 31
    DS_POOLSIZE             : 127

Distribution Cache Entries:
list id   ref  drop hits       last_access          heap_ptr         distribution name
--------------------------------------------------------------------------------


3    0    0    0    42         2020-05-12 10:36:36  4d6b7838
 sysadmin:informix.aus_work_info.aus_info_tabid
3    0    0    0    192        2020-05-12 10:36:36  45c264d8
 sysadmin:informix.aus_work_dist.aus_dist_tabid

6    0    0    0    204        2020-05-12 10:36:36  45c268d8
 sysadmin:informix.aus_work_icols.aus_icols_tabid



Total number of distribution entries: 58
    Number of entries in use        : 0
```

Output description

Number of lists

Number of lists in the distribution cache

DS_POOLSIZE

Number of entries that can be cached at one time

Number of entries

Number of entries in the distribution cache

Number of entries in use

Number of entries that are being used

list

list#

Distribution cache hash chain ID

id

Number of hash entries

ref

ref_cnt

Number of statements that reference a cache entry

drop

dropped?

Whether this entry was dropped after it was added to the cache

hits

The number of times the cache entry is accessed.

last_access

The time at which the cache entry was last accessed.

heap_ptr

Heap address that is used to store this entry

distribution name

The name of the distribution in the cache

Total number of distribution entries

Number of entries in the distribution cache

Number of entries in use

Number of entries that are being used

onstat -g dsk command: Print the progress of the currently running compression operation

Use the onstat -g dsk command to print information that shows the progress of currently running compression operations, such as compress, repack, and shrink.

Syntax:

```
onstat  -g dsk
```

Example output

Figure 117. onstat -g dsk command output for a compress operation

```
                     Processed         Remaining Duration Remaining
Partnum    OP        Rows       Blobs  Rows      Time(s)  Time(s)  Table Name
400002     Compress  6325       1752   1497      00:00:00 00:00:00 db:sl:t1
```

Figure 118. onstat -g dsk command output for a repack operation

```
                         Processed         Remaining Duration Remaining
Partnum   OP      Pass   Rows       Blobs  Rows      Time(s)  Time(s)  Table Name
400002    Repack 1       6325       1752   1497      00:00:00 00:00:00 db:sl:t1
```

Figure 119. onstat -g dsk command output for a compress operation

```
                          Rows            Approx               Approx
Partnum     OP       Processed  Cur Page    Prog  Duration  Remaining  Table
0x00100196  COMPRESS     63253      3515     75%  00:00:03   00:00:01  footab
```

Figure 120. onstat -g dsk command output for a repack operation

```
                         Rows            Approx               Approx
Partnum     OP      Processed  Cur Page    Prog  Duration  Remaining  Table
0x00100196  REPACK      22900      4285     28%  00:00:01   00:00:02  footab
```

Output description

partnum

Partition number of the table or fragment

OP

Compression operation, such as compress, repack, or shrink.

Pass

For repack operations, 1 indicates the first pass of reading the rows, and 2 indicates the second pass.

Processed RowsProcessed

Number of rows that are processed so far for the specified operation

Blobs

The number of simple large objects that were operated on

Remaining Rows

The number of remaining rows to process. For repack operations, the number of rows that remain in the current pass.

Duration Time(s)

The amount of time since the beginning of the operation

Remaining Time(s)

Approximate amount of remaining time for the operation. For repack operations, the amount of time that remains for the current pass.

Curr Page

The current page number that the server is operating on

Approx Prog

Percentage of the total operation that has completed

Duration

The number of seconds that have elapsed since the operation started

Approx Remaining

The approximate time that remains before the operation is complete

Table NameTable

Name of the table

onstat -g env command: Print environment variable values

Use the onstat -g env command to display the values of the environment variables that the database server currently uses.

Syntax:

`onstat -genv` [{ `all` | *variable* | *sessionid [variable]* }]

You can specify one of the following invocations.

Invocation	Explanation
onstat -g env	Displays the settings of environment variables when the database server was started Does not display environment variables that have not been set explicitly.
onstat -g env all	Displays the settings used by all sessions This display is the same as the output of onstat -g env and onstat -g env***sessionid*** iteratively on all current sessions.
onstat -g env ***variable***	Displays the default value of the specified environment variable This ***variable*** argument eliminates the need to pipe the output to **grep** (or some other utility) to locate an environment variable among many that might be set.
onstat -g env ***sessionid***	Displays the settings that a specific session uses. This display includes the following values: • Set in the environment of the session • Assigned by the database server, as **onstat -g env** displays
onstat -g env ***sessionid variable***	Displays the value of the specified environment variable that the specified session uses The ***sessionid*** and ***variable*** arguments eliminate the need to pipe the output to **grep** (or some other utility) to locate an environment variable among many that might be set.

The onstat -g env command displays the current setting of an environment variable and the complete list of values each time the variable was set in the environment. For example, if PDQPRIORITY is set to `10` in the **.onedb.rc** file and set to `55` in the shell environment, onstat -g env command displays both values.

However, if you change the PDQPRIORITY with the onmode -q pdqpriority ***sessionid*** command, the onstat -g env command does not display the new value for the session. The onstat -g env command displays only the values of environment variables set in the environment. It does not display values modified while the session is running.

You might want to display the values of environment variables in the following situations:

- The database server instance has been up for months, and you cannot remember the setting of an environment variable (such as the server locale setting **SERVER_LOCALE**).
- You want to display the complete list of values for an environment variable to identify when an environment variable has been set in multiple places.
- Environment files on disk might have changed or been lost in the interim.
- A support engineer wants to know settings of specific environment variables.

Example output

The following figure shows the output for the onstat -g env command.

Figure 122. onstat -g env command output

```
Variable            Value [values-list]
DBDATE              DMY4/
DBDELIMITER         |
DBPATH              .
DBPRINT             lp -s
DBTEMP              /tmp
ONEDB_HOME         /build2/11.50/tristarm/sqldist
                    [/build2/11.50/tristarm/sqldist]
                    [/usr/informix]
ONEDB_SERVER      parata1150
ONEDB_ TERM        termcap
LANG                C
LC_COLLATE          C
LC_CTYPE            C
LC_MONETARY         C
LC_NUMERIC          C
LC_TIME             C
LD_LIBRARY_PATH     /usr/openwin/lib:/lib:/usr/lib
LKNOTIFY            yes
LOCKDOWN            no
NODEFDAC            no
NON_M6_ATTRS_OK     1
PATH                /build2/11.50/tristarm/sqldist/bin:.:
                    /root/bin:/opt/SUNWspro/bin:/usr/ccs/bin:
                    /usr/openwin/bin:/usr/sbin:/usr/bin:/usr
                    /local/bin
SERVER_LOCALE       en_US.819
SHELL               /bin/ksh
SINGLELEVEL         no
SUBQCACHESZ         10
TBCONFIG            onconfig
TERM                xterm
                    [xterm]
                    [dumb]
TERMCAP             /etc/termcap
TZ                  GB
```

onstat -g ffr command: Print free fragments

Use the onstat -g ffr command to display information about the free memory fragments for a specified session or shared-memory pool.

This command requires an additional argument to specify either a pool name or session ID whose memory pool information is to be displayed. Each session is allocated a memory pool with the same name as the session ID. Use the onstat -g mem command to identify the pool name and the onstat -g ses command to identify the session ID.

Syntax:

onstat -gffr { *pool name* | **sessionid** }

Example output

Figure 124. onstat -g ffr aio command output

```
Free lists for pool name aio:
addr      size      idx
165dcfa0  96        10
1659cf68  152       17
165b2f20  224       26
165c7f20  224       26
1666ec38  968       79
149f2ba0  1120      84
```

Output description

addr (hexadecimal)

Memory address of the pool fragment.

size (decimal)

Size, in bytes, of the pool fragment.

idx (decimal)

For internal use. Index in the array of free list pointers.

onstat -g glo command: Print global multithreading information

Use the onstat -g glo command to display global information about multithreading, information about each virtual processor that is running, and cumulative statistics for each virtual-processor class. This information includes CPU use information about the virtual processors, the total number of sessions, and other multithreading global counters.

Syntax:

```
onstat  -gglo
```

Example output

Figure 126. onstat -g glo command output

```
MT global info:
sessions threads  vps      lngspins time
0        23       14       0        142

          sched calls     thread switches yield 0   yield n   yield forever
total:    85240           70451           16956     868       37319
per sec:  0               0               0         0         0

Virtual processor summary:
 class       vps       usercpu   syscpu    total
 cpu         1         92.12     0.59      92.71
 aio         1         0.05      0.08      0.13
 lio         1         0.00      0.00      0.00
 pio         1         0.00      0.00      0.00
 adm         1         0.00      0.01      0.01
 soc         4         0.01      0.01      0.02
 msc         1         0.00      0.00      0.00
 jvp         1         0.00      0.00      0.00
 fifo        1         0.00      0.00      0.00
 nyevp       1         0.00      0.00      0.00
 yevp        1         0.00      0.00      0.00
 total       14        92.18     0.69      92.87

Individual virtual processors:
 vp    pid       class       usercpu   syscpu    total     Thread    Eff
 1     26328     cpu         92.12     0.59      92.71     122.65    75%
 2     26330     adm         0.00      0.01      0.01      0.00       0%
 3     26331     lio         0.00      0.00      0.00      0.00       0%
 4     26332     pio         0.00      0.00      0.00      0.00       0%
 5     26333     aio         0.05      0.08      0.13      0.28      45%
 6     26334     msc         0.00      0.00      0.00      0.19       0%
 7     26335     fifo        0.00      0.00      0.00      0.00       0%
 8     26336     nyevp       0.00      0.00      0.00      0.00       0%
 9     26337     yevp        0.00      0.00      0.00      0.00       0%
 10    26338     jvp         0.00      0.00      0.00      0.00       0%
 11    26339     soc         0.00      0.00      0.00      NA         NA
 12    26340     soc         0.00      0.00      0.00      NA         NA
 13    26341     soc         0.01      0.01      0.02      NA         NA
 14    26342     soc         0.00      0.00      0.00      NA         NA
                 tot         92.18     0.69      92.87
```

```
MT global info:
sessions threads  vps      lngspins
0        28       11       0

          sched calls     thread switches  yield 0   yield n   yield forever
total:    19577737        16256991         3688561   11340793  280099
per sec:  0               0                 0         0         0

Virtual processor summary:
 class       vps       usercpu   syscpu    total
 cpu         1         107.40    8.09      115.49
 aio         6         4.19      94.42     98.61
 lio         1         0.68      3.54      4.22
 pio         1         0.03      0.33      0.36
 adm         1         0.00      0.05      0.05
 msc         1         0.00      0.01      0.01
 total       11        112.30    106.44    218.74

Individual virtual processors:
```

Output description

The following table explains each column in the global information section of the example output.

Table 155. Description of the columns in the virtual processor summary

Column name	Description
sessions	The number of sessions
threads	The total number of threads
vps	The total number of virtual processors
lngspins	The number of times a thread had to spin more than 10,000 times to acquire a latch on a resource
time	The number of seconds over which the statistics were gathered. Statistics start when the server starts or the statistics are reset by running the onstat -z command.
sched calls	The total number of scheduled calls.
thread switches	The total number of switches from one thread to another.
yield	Statistics on thread yields, which occur when a thread can no longer continue its task until some condition occurs

The following table explains each column in the virtual processor summary section of the example output.

Table 156. Description of the columns in the virtual processor summary

Column name	Description
class	The type of virtual processor.
vps	The number of instances of the class of virtual processor.
usercpu	The total user time, in seconds, that the class of virtual processor spent running on the CPU.
syscpu	The total system time, in seconds, the class of virtual processor spent running on the CPU.
total	The total CPU time for the virtual processor class, as the sum of the user time plus the system time.

The following table explains each column in the individual virtual processors section of the example output.

Table 157. Description of the columns for the individual virtual processors

Column name	Description
vp	The virtual processor number. On Windows™, the values are thread IDs.
pid	The Process ID of the oninit process.

Table 157. Description of the columns for the individual virtual processors (continued)

Column name	Description
class	The type of virtual processor.
usercpu	The total user time, in seconds, that the virtual processor spent running on the CPU.
syscpu	The total system time, in seconds, that the virtual processor spent running on the CPU.
total	The total CPU time for the virtual processor, as the sum of the user time plus the system time.
Thread	The total time the threads ran on the virtual processor.
Eff	Efficiency. The ratio of the total CPU time to the total time the threads ran on the virtual processor.

onstat -g his command: Print SQL trace information

Use the onstat -g his command to display SQL trace information from the collection of **syssqltrace** tables (syssqltrace, syssqltrace_info, syssqltrace_hvar and syssqltrace_itr) in the **sysmaster** database.

The level setting of the SQLTRACE configuration parameter affects what SQL trace information is stored and displayed by the set of **syssqltrace** tables, and what information onstat -g his displays. Each row of the **syssqltrace** table describes a previously executed SQL statement. By default, only the DBSA can view the **syssqltrace** information from the onstat -g his command. However, when the UNSECURE_ONSTAT configuration parameter is set to `1`, all users can view this information.

Syntax:

```
onstat  -ghis
```

Example output

The content of the output depends on the trace settings.

The **Statement history** section in the output provides information about the current settings for tracing.

```
Statement history:

Trace Level                  Low
Trace Mode                Global
Number of traces            1000
Current Stmt ID                2
Trace Buffer size           2008
Duration of buffer           293  Seconds
Trace Flags           0x00001611
Control Block         0x4c2f0028
```

The following table describes this output:

Information	Description
Trace Level	Amount of information traced. Valid values are LOW, MED, HIGH, and OFF.
Trace Mode	Type of tracing performed. Global refers to all users on the system User refers to only those users who have tracing enabled by an SQL administration API function.
Number of traces	The number of SQL statements that are being traced. This is the value set in your onconfig file unless the **ntraces** parameter is changed dynamically through SQL Administration API functions. The range is 500 to 2147483647. If you have 100,000 trace buffers and your organization runs 1000 SQL statements a second, and are tracing all of the statements, then the buffers would last for 100 seconds before they would begin being overwritten.
Current® Stmt ID	The ID for the current SQL statement. Each statement being traced gets a unique ID.
Trace Buffer size	The amount of data each trace buffer will capture, in bytes. If you set the size to 2KB, but have an SQL statement that is 12KB, the statement is truncated by at least 10KB. More data might be truncated, depending on what else is being traced.
Duration of buffer	The amount of time, in seconds, that the trace data in the current trace buffer spans. This is not how long the sqltrace feature has been running. In the above example **Duration of buffer** is 293 seconds which indicates the number of seconds between the first and last SQL statement that are traced.
Trace Flags	The current SQL trace flags that are set.
Control Block	The memory address of the SQL trace control block.

The information displayed below is repeated one time for each time a statement was run. In this example there are two variables being called.

```
Statement # 2:     @ 0x4c2f3028

Database:        sysmaster
 Statement text:
  select count(*) from systables,syscolumns where systables.tabid > ? and
    systables.nrows < ?

  SELECT using tables [ systables syscolumns ]
```

The following table describes this output:

Information	Description
Database	The name of the database or part number of the **systables** entry for the database.
Statement text	The statement text for this SQL statement. If the statement is a stored procedure, then the statement text would display the procedure stack trace. The statement text might be truncated if the statement and the numeric statistics are larger than the trace buffer.

```
Iterator/Explain
================
ID   Left  Right   Est Cost   Est Rows   Num Rows   Partnum   Type
 3      0      0         17         42        146   1048579   Index Scan
 4      0      0       5249       2366       2366   1048580   Seq Scan
 2      3      4       5266      99372     345436         0   Nested Join
 1      2      0          1          1          1         0   Group
```

The following table describes this output:

Information	Description
ID	SQL iterator ID
Left	ID of the left input to the iterator
Right	ID of the right input to the iterator
Est Cost	Estimated cost of this iterator
Est Rows	Estimated rows for this iterator
Num Rows	Actual number of rows for this iterator
Partnum	The table or index partition number.
Type	Type of operation

If the SQL statement contains one or more variables, and you are tracing host variables, the **Host Variables** section is included in the output.

```
Host Variables
==============
1 integer    100
2 float      1000.0000000000000000
```

The following table describes this output:

Information	Description
Column 1	The position of the variable in the statement.
Column 2	The data type of the variable.
Column 3	The value of the variable.

```
Statement information:
  Sess_id  User_id  Stmt Type  Finish Time   Run Time   TX Stamp  PDQ
  5        2053     SELECT     01:08:48      0.4247     340a6e9    0
```

The following table describes this output:

Information	Description
Sess_id	The session ID
User_id	The operating system user ID
Stmt Type	The type of SQL statement
Finish Time	The time of day that the SQL statement finished
Run Time	The total amount of time consumed by the virtual processors or threads used to process the statement. For example, if the Finish Time is 1:15:00 and the Run Time is 9 minutes and the start time is not necessarily 1:06:00. There might be multiple virtual processors or threads involved in processing parts of the statement in parallel.
TX Stamp	The time the BEGIN WORK statement was logged in this transaction
PDQ	The SQL statement PDQ level

The **Statement Statistics** section in the output provides specific information about the statement.

```
  Statement Statistics:
  Page       Buffer     Read       Buffer     Page       Buffer     Write
  Read       Read       % Cache    IDX Read   Write      Write      % Cache
  1285       19444      93.39      0          810        17046      95.25

  Lock       Lock       LK Wait    Log        Num        Disk       Memory
  Requests   Waits      Time (S)   Space      Sorts      Sorts      Sorts
  10603      0          0.0000     60.4 KB    0          0          0

  Total      Total      Avg        Max        Avg        I/O Wait   Avg Rows
  Executions Time (S)   Time (S)   Time (S)   IO Wait    Time (S)   Per Sec
  1          30.8660    30.8660    30.8660    0.0141     29.2329    169.8959

  Estimated  Estimated  Actual     SQL        ISAM       Isolation  SQL
  Cost       Rows       Rows       Error      Error      Level      Memory
```

```
102        1376       5244       0          0          CR         32608
```

Information	Description
Page Read	Number of pages that have been read from disk for this SQL statement
Buffer Read	Number of times a page has been read from the buffer pool and not read from disk for this SQL statement
Read % Cache	Percentage of times the page was read from the buffer pool
Buffer IDX Read	This Currently not implemented
Page Write	Number of pages written to disk
Buffer Write	Number of pages modified and sent back to the buffer pool
Write % Cache	Percentage of time that a page was written to the buffer pool but not to disk
Lock Requests	Total number of locks required by this statement
Lock Waits	Number of times this SQL statement waited on locks
LK Wait Time (S)	Amount of time the statement waited for application locks, in seconds
Log Space	Amount of storage space that the SQL statement used in the logical log
Num Sorts	Total number of sorts used to execute the statement
Disk Sorts	Number of sorts which required disk space to execute the sort for this SQL statement
Memory Sorts	Number of sorts executed which executed entirely in memory for this SQL statement
Total Executions	Total number of times this prepared statement has been executed, or the number of times this cursor has been re-used
Total Time (S)	Total time this prepared statement ran, in seconds
Avg Time (S)	Average time this prepared statement required to execute, in seconds
Max Time (S)	Total time to run the prepared SQL statement, in seconds, excluding any time taken by the application. If you prepare a query then run the query 5 times, each time the query is run a trace is added to the trace buffer. The Max Time is the maximum time any one execution took.

Information	Description
Avg IO Wait	Average amount of time the statement waited for I/O, excluding any asynchronous I/O
I/O Wait Time (S)	Amount of time the statement waited for I/O, excluding any asynchronous I/O, in seconds
Avg Rows Per Sec	Average number of rows a second produced by this statement
Estimated Cost	The query optimizer cost associated with the SQL statement
Estimated Rows	Number of rows returned by the statement, as estimated by the query optimizer
Actual Rows	Number of rows returned for this statement
SQL Error	The SQL error number
ISAM Error	The RSAM or ISAM error number
Isolation Level	Isolation level this statement was run with
SQL Memory	Number of bytes this SQL statement required

For the complete schema of the **syssqltrace** System Monitoring Interface table, see syssqltrace on page 272.

For details of setting the SQLTRACE configuration parameter, see SQLTRACE configuration parameter on page 189.

onstat -g ioa command: Print combined onstat -g information

Use the onstat -g ioa command to display combined information from the onstat -g iob, onstat -g iof, onstat -g ioq, and onstat -g iov commands.

Syntax

```
onstat  -gioa
```

Example output

```
AIO global info:
   9 aio classes
   9 open files
  64 max global files

AIO I/O queues:
q name/id    len maxlen totalops  dskread dskwrite  dskcopy
 fifo     0    0      0        0        0        0        0
drda_dbg 0    0      0        0        0        0        0
sqli_dbg 0    0      0        0        0        0        0
  adt     0    0      0        0        0        0        0
  msc     0    0      1      231        0        0        0
```

```
  aio    0    0      5    13069    10895        0        0
  pio    0    0      1     1580        0     1580        0
  lio    0    0      1    37900        0    37900        0
  gfd    3    0     87    42115    15806    26309        0
  gfd    4    0      4        5        1        4        0
  gfd    5    0     12       35       22       13        0
  gfd    6    0     11       33       21       12        0
  gfd    7    0      1        4        3        1        0
  gfd    8    0      1        4        3        1        0

AIO I/O vps:
class/vp/id s  io/s totalops dskread dskwrite dskcopy wakeups io/wup errors tempops
 fifo  7  0 i   0.0        0       0        0       0       1  0.0      0        0
  msc  6  0 i   0.0      231       0        0       0     221  1.0      0      231
  aio  5  0 i   0.0    39285   26358    10793       0   37531  1.0      0        5
  aio  9  1 i   0.0     5770    3795     1944       0    5926  1.0      0        0
  aio 10  2 i   0.0     2308     717     1585       0    1953  1.2      0        0
  aio 11  3 i   0.0     1463     166     1295       0    1166  1.3      0        0
  aio 12  4 i   0.0     1219      46     1172       0     943  1.3      0        0
  aio 13  5 i   0.0     1041      34     1007       0     805  1.3      0        0
  aio 15  6 i   0.0      425       2      423       0     438  1.0      0        0
  aio 16  7 i   0.0      342       5      337       0     395  0.9      0        0
  pio  4  0 i   0.0     1580       0     1580       0    1581  1.0      0     1580
  lio  3  0 i   0.0    37900       0    37900       0   29940  1.3      0    37900

AIO global files:
gfd pathname         bytes read     page reads  bytes write    page writes io/s
3   ./rootdbs        85456896       41727       207394816      101267      572.9
 op type     count          avg. time
 seeks       0              N/A
 reads       13975          0.0015
 writes      51815          0.0018
 kaio_reads  0              N/A
 kaio_writes 0              N/A

4   tempsbs.chunk    2048           1           8192           4           113.6
 op type     count          avg. time
 seeks       0              N/A
 reads       1              0.0131
 writes      3              0.0074
 kaio_reads  0              N/A
 kaio_writes 0              N/A

5   sbs1.chunk       45056          22          26624          13          173.4
 op type     count          avg. time
 seeks       0              N/A
 reads       22             0.0063
 writes      6              0.0038
 kaio_reads  0              N/A
 kaio_writes 0              N/A

6   sbs2.chunk       43008          21          24576          12          76.1
 op type     count          avg. time
 seeks       0              N/A
 reads       21             0.0148
 writes      6              0.0072
 kaio_reads  0              N/A
 kaio_writes 0              N/A
```

```
7   qhdr.chunk       6144          3           2048           1           550.5
 op type     count          avg. time
 seeks       0              N/A
 reads       3              0.0019
 writes      1              0.0016
 kaio_reads  0              N/A
 kaio_writes 0              N/A

8   ./dbs1           6144          3           2048           1           403.0
 op type     count          avg. time
 seeks       0              N/A
 reads       3              0.0027
 writes      1              0.0018
 kaio_reads  0              N/A
 kaio_writes 0              N/A


AIO big buffer usage summary:
class                 reads                                  writes
       pages     ops  pgs/op  holes  hl-ops hls/op      pages    ops  pgs/op
fifo       0       0   0.00      0       0   0.00           0      0   0.00
drda_dbg   0       0   0.00      0       0   0.00           0      0   0.00
sqli_dbg   0       0   0.00      0       0   0.00           0      0   0.00
 kio       0       0   0.00      0       0   0.00           0      0   0.00
 adt       0       0   0.00      0       0   0.00           0      0   0.00
 msc       0       0   0.00      0       0   0.00           0      0   0.00
 aio  228709   20228  11.31   1005     203   4.95      213272  18556  11.49
 pio       0       0   0.00      0       0   0.00       19672   1580  12.45
 lio       0       0   0.00      0       0   0.00       55287  37900   1.46
```

Output description

For a description of each output column, see the individual onstat -g iob command: Print big buffer use summary on page 570, onstat -g ioq command: Print I/O queue information on page 572, and onstat -g iov command: Print AIO VP statistics on page 575 commands.

onstat -g iob command: Print big buffer use summary

Use the onstat -g iob command to display a summary of big buffer use.

Syntax:

```
onstat  -giob
```

Example output

Figure 130. onstat -g iob command output

```
AIO big buffer usage summary:
                 reads                                       writes
       pages     ops   pgs/op   holes   hl-ops hls/op      pages     ops   pgs/op
fifo      0        0    0.00      0        0    0.00          0       0    0.00
 kio      0        0    0.00      0        0    0.00          0       0    0.00
 adt      0        0    0.00      0        0    0.00          0       0    0.00
 msc      0        0    0.00      0        0    0.00          0       0    0.00
 aio      0        0    0.00      0        0    0.00        607     607    1.00
 pio      0        0    0.00      0        0    0.00          0       0    0.00
 lio      0        0    0.00      0        0    0.00          0       0    0.00
```

onstat -g iof command: Print asynchronous I/O statistics

Use the onstat -g iof command to display the asynchronous I/O statistics by chunk or file.

This command is similar to the onstat -D command, except that onstat -g iof also displays information on nonchunk files. It includes information about temporary files and sort-work files.

Syntax:

```
onstat  -giof
```

Example output

Figure 132. onstat -g iof command output

```
AIO global files:
gfd pathname      bytes read    page reads    bytes write    page writes   io/s
3   rootdbs       1918976       937           145061888      70831         36.5

      op type     count          avg. time
      seeks       0              N/A
      reads       937            0.0010
      writes      4088           0.0335
      kaio_reads  0              N/A
      kaio_writes 0              N/A
```

Output description

gfd

Global file descriptor number for this chunk or file.

pathname

The pathname of the chunk or file.

bytes read

Number of byte reads that have occurred against the chunk or file.

page reads

Number of page reads that have occurred against the chunk or file.

bytes write

Number of byte writes that have occurred against the chunk or file.

page writes

Number of page writes that have occurred against the chunk or file.

io/s

Number of I/O operations that can be performed per second. This value represents the I/O performance of the chunk or file.

op type

Type of operation.

count

Number of times the operation occurred.

avg time

Average time the operation took to complete.

onstat -g iog command: Print AIO global information

Use the onstat -g iog command to display global information about AIO.

Syntax:

```
onstat  -giog
```

Example output

Figure 134. onstat -g iog command output

```
AIO global info:
   8 aio es
   5 open files
  64 max global files
```

onstat -g ioq command: Print I/O queue information

Use the onstat -g ioq command to display statistics about the number and types of operations performed by I/O queues.

Syntax:

onstat -gioq [*queue_name*]

If a *queue_name* is given then only queues with that name are shown. If no *queue_name* is given then information is given for all queues.

Example output

Figure 136. onstat -g ioq command output

```
AIO I/O queues:
q name/id    len maxlen totalops  dskread dskwrite  dskcopy
sqli_dbg   0      0      0        0        0        0        0
 fifo   0      0      0        0        0        0        0
  adt   0      0      0        0        0        0        0
  msc   0      0      1      537        0        0        0
  aio   0      0      3     6537      238     5777        0
  pio   0      0      2     1103        0     1102        0
  lio   0      0      2    11795        0    11794        0
  gfd   3      0     17    17489     1526    15963        0
  gfd   4      0     17    18347     2384    15963        0
  gfd   5      0     16      220       41      179        0
  gfd   6      0      4        4        0        4        0
  gfd   7      0      4        4        0        4        0
  gfd   8      0      4        4        0        4        0
  gfd   9      0      9       54       24       30        0
  gfd  10      0     16      149       40      109        0
  gfd  11      0     16      621      128      493        0
  gfd  12      0     16     1953     1146      807        0
  gfd  13      0     16      409       71      338        0
  gfd  14      0     16      378       60      318        0
```

Output description

q name/id

The name and number of the I/O queue. The name indicates what type of queue it is. The number is used to tell queues of the same name apart.

Here is a list of the possible queue names and what each type of queue handles:

sqli_dbg

Handles I/O for HCL Technical Support's SQL Interface Debugging feature

fifo

Handles I/O for FIFO VPs

adt

Handles auditing I/O

msc

Handles miscellaneous I/O

aio

Handles HCL OneDB™ asynchronous I/O

kio

Handles kernel AIO

pio

Handles physical logging I/O

lio

Handles logical logging I/O

gfd

Global File Descriptor - Each primary and mirror chunk is given a separate global file descriptor. Individual gfd queues are used depending on whether kaio is on and the associated chunk is cooked or raw.

len

The number of pending I/O requests in the queue

maxlen

The largest number of I/O requests that have been in the queue at the same time

totalops

The total number of I/O operations that have been completed for the queue

dskread

Total number of completed read operations for the queue

dskwrite

Total number of completed write operations for the queue

dskcopy

Total number of completed copy operations for the queue

onstat -g ipl command: Print index page logging status information

Use the onstat -g ipl command to display information about the status of index page logging.

Syntax:

```
onstat -gipl
```

Example output

Figure 138. onstat -g ipl command output

```
Index page logging status: Enabled
Index page logging was enabled at: 2008/12/20 16:01:02
```

Output description

Index page logging status

Status of index page logging: Enabled or Disabled.

Index page logging was enabled at

The date and time at which index page logging was enabled.

onstat -g iov command: Print AIO VP statistics

Use the onstat -g iov command to display asynchronous I/O statistics for each virtual processor.

Syntax:

```
onstat  -giov
```

Example output

Figure 140. onstat -g iov command output

```
AIO I/O vps:
class/vp/id  s  io/s  totalops  dskread dskwrite  dskcopy  wakeups  io/wup  errors tempops
fifo  7  0   i   0.0        0        0        0        0        1     0.0      0       0
msc   6  0   i   0.1     9988        0        0        0     7833     1.3      0    9988
aio   5  0   i   0.0     4894     3341     1426        0     4393     1.1      0       0
aio   9  1   i   0.0       41        0       41        0       33     1.2      0       0
pio   4  0   i   0.0      199        0      199        0      200     1.0      0     199
lio   3  0   i   0.0     6344        0     6344        0     6344     1.0      0    6344
```

Output description

class

The class of the virtual processor.

vp

The ID number of the virtual processor within its class.

s

Current® status of the AIO virtual processor

f

Fork

i

Idle

s

Search

b

Busy

o

Open

c

Close

io/s

The average I/O speed (measured in operations per second) for the virtual processor since the time the database server started or since the onstat -z command was last run, whichever happened last.

totalops

Total number of I/O operations performed by this virtual processor since the time the database server started or since the onstat -z command was last run, whichever happened last.

dskread

Total number of read operations performed by this virtual processor since the time the database server started or since the onstat -z command was last run, whichever happened last.

dskwrite

Total number of write operations performed by this virtual processor since the time the database server started or since the onstat -z command was last run, whichever happened last.

dskcopy

Total number of copy operations performed by this virtual processor since the time the database server started or since the onstat -z command was last run, whichever happened last.

wakeups

For AIO VPs, the number of times the virtual processor has gone idle since the time the database server started or since the onstat -z command was last run, whichever happened last.

io/wup

For AIO VPs, the average number of I/O operations performed per wake-up by this virtual processor since the time the database server started or since the onstat -z command was last run, whichever happened last.

errors

Total number of KAIO out of resource errors.

tempops (decimal)

For internal use only. This is I/O operation counter that is maintained to determine when a new AIO VP should be added. It is applicable only when the AUTO_AIOVPS configuration parameter is enabled.

onstat -g lap command: Print light appends status information

Use the onstat -g lap command to display information about the status of light appends occurring in the system.

Syntax:

```
onstat -glap
```

Example output

Figure 142. onstat -g lap command output

```
Light Append Info
session id  address   cur_ppage  la_npused  la_ndata   la_nrows   bufcnt
31          b60a5e8   ffbff494   2938       2937       93990      4
```

Output description

Session id (decimal)

Session ID performing the light append operation

address (hexadecimal)

Address of the light append buffer

cur_ppage (hexadecimal)

Current® physical page address

la_npused (decimal)

Number of pages allocated

la_ndata (decimal)

Number of data pages appended

la_nrows (decimal)

Number of rows appended

bufcnt (decimal)

Number of light append buffers

onstat -g laq command: Print log apply queues

Use the onstat -g laq command to print information about log recovery apply queues.

Use the onstat -g laq command to print information about log recovery apply queues. This includes logical log recovery on secondary servers as well as logical restore or logical recovery part of fast recovery. Log records from logical logs are assigned to replay worker threads according to the tablespace ID (partnum) associated with them; a subset of log records will be applied by the replay master thread.

For instance, in a high-availability cluster, the primary server sends log records to one or more secondary servers over the network. Each secondary server continuously replays the transaction logs from the primary server to ensure that data is replicated on the secondary server. Each tblspace on the primary server is assigned a queue on the secondary server in which to receive log records. A replay thread applies the log records stored in the queue to the secondary server. The log records are applied in the order in which they were received.

You use the onstat -g laq command to monitor the performance of the log apply queues, on a secondary server or during any other form of log recovery. Use this command if you suspect that the primary server performance is slowed because logs are not replaying quickly enough on the secondary server, or to monitor the progress made during logical restore. The Avg Depth (average depth) column indicates the average number of log records in the queue(Queue Size) incurred whenever putting a new log record on a queue. The Current/Last LSN column specifies the log record a replay thread currently is active on, or the last one it was replaying, with the Partval column typically specifying the tablespace ID this log record refers to. Transaction pointer and ID shown for a replay thread indicate a log record currently being applied.

When used in repeat mode, using -r [<seconds>][.<fraction>] option, an overall log record apply rate is calculated and shown.

On a secondary server, the transaction latency measured, in full seconds, with each end-of-transaction(COMMIT, ROLLBACK) log record as difference between local apply time and primary server's EoT time, is shown.

The onstat -g laq command is valid only when some form of logical log recovery is going on, otherwise only an onstat header is printed.

Syntax:

```
onstat -g laq
```

Example output

Figure 144. onstat -g laq -r .3 command output from a remote standalone secondary server

```
Log Apply Info:
Thread          Queue        Total      Avg
                 Size       Queued    Depth   Current/Last LSN    Partval Txp      (Txid)
wreplay_1           1       938310    19.66     14087,0x482ec      100540 0x450a8c28 (29)
wreplay_2           0       782865    12.91     14087,0x48184
wreplay_3           3       937766    19.86     14087,0x45598      100542 0x450a8c28 (29)
wreplay_4           2       529755    14.43     14087,0x483bc      100543 0x450a8c28 (29)
wreplay_5           6       389432    10.93     14087,0x46500      10054e 0x450a8c28 (29)
wreplay_6           0       789238    10.90     14087,0x4318c
wreplay_7           0      1317820    20.26     14087,0x440b0
wreplay_8           0       991836    12.29     14086,0xb3d8
wreplay_9           0       851854    19.60     14086,0xb52c
wreplay_10          1       913434     9.42     14087,0x4849c          1d 0x450a8c28 (29)
mreplay                     689544              14087,0x4849c
Total:             13      9131854   150.26  Avg:    15.03

Secondary Apply Queue:         Total Buffers:12 Size:1024K Free Buffers:11
Log Recovery Queue:            Total Buffers:12 Size:16K Free Buffers:10
Log Page Queue:                Total Buffers:512 Size:2K Free Buffers:512
Log Record Queue:              Total Buffers:50 Size:16K Free Buffers:42

Transaction Latency: 1 seconds
Apply rate:  30213.33 recs/sec - 9064 new recs in 300ms
```

Output description

Thread

The name of the apply thread for a given log record queue.

Queue Size

The number of log records queued for a given apply thread.

Total Queued

The total number of queued log records for a given apply thread.

Avg Depth

The average number of log records in the queue at the time a queue insert operation occurred.

Secondary Apply Queue

The secondary apply queue receives log buffers from the primary server. The values displayed represent the total number of buffers allocated to receiving log buffer records(SEC_DR_BUFS), the size of the buffers(LOGBUFF), and the number of currently unused buffers.

Log Recovery Queue

The log recovery queue receives output from the secondary apply queue. The log buffers are converted to a format compatible with the ontape utility. The values displayed represent the total number of stream buffers in the recovery queue, the size of the stream buffers(LTAPEBLK), and the number of unused buffers.

Log Page Queue

The log page queue receives output from the log recovery queue. The values displayed represent the total number of log pages in the queue, the size of the queue, and the number of unused buffers.

Log Record Queue

The log record queue receives output from the log page queue. The log pages are divided into individual log records. The values displayed represent the total number of log records in the recovery queue, the size of the queue, and the number of unused buffers.

Transaction Latency

Time difference between last replayed transaction commit time at primary server and local server. For this to be accurate, operating system time must match between primary and secondary servers.

Apply rate

Number of log records replayed per second. Apply rate is only shown with -r option.

onstat -g lmm command: Print low memory management information

Use the onstat -g lmm command to display information about automatic low memory management settings and recent activity.

Syntax:

```
onstat  -g lmm
```

Example output

Figure 146. onstat -g lmm command output

```
Low Memory Manager

Control Block      0x4cfca220
Memory Limit       300000 KB
Used               149952 KB
Start Threshold    10240 KB
Stop Threshold     10 MB
Idle Time          300 Sec
Internal Task      Yes
Task Name         ‘Low Memory Manager’
Low Mem TID        0x4cfd7178
# Extra Segments   0

Low Memory Manager Tasks

Task                Count    Last Run
Kill User Sessions  267      04/04/2011.16:57
Kill All Sessions   1        04/04/2011.16:58
Reconfig(reduce)    1        04/04/2011.16:59
Reconfig(restore)   1        04/04/2011.17:59

Last 20 Sessions Killed

Ses ID Username Hostname PID   Time
194    sfisher  host01   13433 04/04/2011.16:57
201    sfisher  host01   13394 04/04/2011.16:57
198    sfisher  host01   13419 04/04/2011.16:57
190    sfisher  host01   13402 04/04/2011.16:57
199    sfisher  host01   13431 04/04/2011.16:57

Total Killed 177
```

Output description

Control Block

Address of the internal control structure for automatic low memory management

Memory Limit

Amount of memory to which the server is attempting to adhere

Used

Amount of memory currently used by the server

Start Threshold

Value for the automatic low memory management start threshold

Stop Threshold

Value for the automatic low memory management stop threshold

Idle Time

The amount of time after which automatic low memory management considers a session idle

Internal Task

Yes = using HCL OneDB™ procedures

No = using user-defined procedures

Task Name

Name of user-defined procedure

Low Mem TID

Address of the automatic low memory management thread

Task

Kill = Automatic processes ran and terminated sessions.

Reconfig(reduce) = Automatic processes ran and freed blocks of unused memory.

Reconfig(restore) = Automatic processes ran and restored services and configuration.

Count

Number of times that the task ran

Last Run

Date and time when the last task ran

Ses ID

ID of session that was terminated (with an onmode –z command)

Username

User name of the owner of the session

Hostname

Name of the host where the session originated

PID

Process ID

Time

Date and time when the session was terminated

You use the LOW_MEMORY_MGR configuration parameter to enable the automatic low memory management.

onstat -g lmx command: Print all locked mutexes

Use the onstat -g lmx command to display information about all locked mutexes.

Syntax:

```
onstat -glmx
```

Example output

Figure 148. onstat -g lmx command output

```
Locked mutexes:
mid      addr              name              holder   lkcnt  waiter   waittime
119006   7000001e684b928   td_mutex          298      0
134825   7000002043a9148   free_lock         11009    0      200      22921
                                                             11010    22918
587817   70000022ddb3268   sync_lock1        200      0
593614   700000239ce7b68   SB_LTH_LATCH      875      0

Number of mutexes on VP free lists: 49
```

```
Locked mutexes:
mid     addr        name       holder  lkcnt  waiter   waittime
3258    17389dc8    nsf.lock   1313    0      13       491
                                              1184     188
Number of mutexes on VP free lists: 49
```

Output description

mid

Internal mutex identifier

addr

Address of locked mutex

name

Name of the mutex

holder

Thread ID of the thread that is holding the mutex

0 = The read/write mutex is held in shared mode

lkcnt

For a read/write mutex, the current number of threads that are locking the mutex in shared mode. For a relockable mutex, the number of times the mutex was locked or relocked by the thread that is holding the mutex.

The number of times the mutex was locked or relocked by the thread that is holding the mutex

waiter

List of IDs of the threads that are waiting for this mutex

waittime

Amount of time in seconds that the thread is waiting

onstat -g lsc command: Print active light scan status (deprecated)

The onstat -g lsc command has been superseded by the onstat -g scn command.

Syntax:

```
onstat  -glsc
```

Example output

Figure 150. onstat -g lsc command output

```
Light Scan Info
descriptor   address          next_lpage   next_ppage       ppage_left   bufcnt   look_aside
3            474b74b0         4a0          7e2c80           416          1             N
```

Output description

descriptor (decimal)

Light scan ID

address (hex)

Memory address of the light scan descriptor

next_lpage (hex)

Next logical page address to scan

next_ppage (hex)

Next physical page address to scan

ppage_left (decimal)

Number of physical pages left to scan in the current extent

#bufcnt (decimal)

Number of light scan buffers used for this light scan

#look_aside (char)

Whether look aside is needed for this light scan (`Y` = yes, `N` = no). Look asides occur when a thread needs to examine the buffer pool for existing pages to obtain the latest image of a page being light scanned.

Use the onstat -g scn command to display the status of a current scan, based on rows scanned on compressed tables, tables with rows that are larger than a page, and tables with VARCHAR, LVARCHAR, and NVARCHAR data. For more information, see onstat -g scn command: Print scan information on page 626.

onstat -g mem command: Print pool memory statistics

Use the onstat -g mem command to display the memory statistics for a pool.

If you run an SQL query that allocates memory from the PER_STMT_EXEC and PER_STMT_PREP memory duration pools, the onstat -g mem command displays information about the **PRP.sessionid.threadid** pool and the **EXE.sessionid.threadid** pool.

Syntax:

`onstat -gmem` [*pool namesession id*]

Session pools are named with the session number. If no argument is provided, information about all pools is displayed.

Example output

Figure 152. onstat -g mem command output

```
Pool Summary:
name            addr            totalsize freesize #allocfrag #freefrag
resident     R     10a001028       2420736   7960     2          2
res-buff     R     10a250028       8269824   7960     2          2
global       V     10aac0028       9351168   32648    650        11
...
...
...
onmode_mon   V     10b983028       20480     2752     108        1
13           V     10bd5d028       16384     5200     12         2
Blkpool Summary:
name               addr            size      #blks      pre-hint   szavail|
global       V     10aac8920       0         0          0          0
xmf_msc_pl   V     10ac84ca0       954368    73         0          0
```

Output description

Pool Summary

name

Pool name

Shared memory segment type where the pool is created

addr

Pool memory address

totalsize

Pool size, in bytes

freesize

Free memory in pool

#allocfrag

Allocated fragments in pool

#freefrag

Free fragments in pool

Blkpool Summary

name

Pool name

Shared memory segment type where pool is created

addr

Pool memory address

size

Pool size, in bytes

#blks

Number of blocks in pool

onstat -g mgm command: Print MGM resource information

Use the onstat -g mgm command to show resource information about Memory Grant Manager (MGM).

You can use the onstat -g mgm command to monitor how MGM coordinates memory use and scan threads. This command reads shared-memory structures and provides statistics that are accurate at the instant that the command runs.

Syntax:

```
onstat  -gmgm
```

The onstat -g mgm output shows a unit of memory that is called a *quantum*. The *memory quantum* represents a unit of memory, as follows:

```
memory quantum = DS_TOTAL_MEMORY / DS_MAX_QUERIES
```

The following calculation shows the memory quantum for the values that the onstat -g mgm output shows:

```
memory quantum = 4000 kilobytes / 31
               = 129 kilobytes
```

The database server adjusts the value of a quantum as needed when it grants memory. Therefore, the value of the quantum as shown by the onstat -g mgm command is not always accurate.

The *scan thread quantum* is always equal to `1`.

Example output

Figure 154. onstat -g mgm command output

```
Memory Grant Manager (MGM)
--------------------------
MAX_PDQPRIORITY:  100
DS_MAX_QUERIES:    31
DS_MAX_SCANS:      1048576
DS_NONPDQ_QUERY_MEM: 128 KB
DS_TOTAL_MEMORY:   4000 KB

Queries:   Active     Ready   Maximum
                        0         0        31
Memory:     Total      Free   Quantum
(KB)         4000      4000       128

Scans:      Total      Free   Quantum
               1048576   1048576         1

Load Control:    (Memory)      (Scans)  (Priority)  (Max Queries)   (Reinit)
                       Gate 1       Gate 2      Gate 3         Gate 4     Gate 5
(Queue Length)          0            0           0              0          0

Active Queries:  None
Ready Queries:  None
Free Resource        Average #        Minimum #
--------------    ---------------     ---------
Memory               0.0 +- 0.0           500
Scans                0.0 +- 0.0          1048576

Queries              Average #        Maximum #    Total #
--------------    ---------------     ---------    -------
Active               0.0 +- 0.0            0          0
Ready                0.0 +- 0.0            0          0

Resource/Lock Cycle Prevention count:  0
```

Output description

The first portion of the output shows the values of the PDQ configuration parameters.

The second portion of the output describes MGM internal control information. It includes four groups of information. The first group is **Queries**:

Active

Number of PDQ queries that are currently running

Ready

Number of user queries ready to run but whose execution the database server deferred for load-control reason

Maximum

Maximum number of queries that the database server allows to be active. Reflects current value of the DS_MAX_QUERIES configuration parameter

The next group is **Memory**:

Total

KB of memory available for use by PDQ queries (DS_TOTAL_MEMORY specifies this value.)

Free

KB of memory for PDQ queries not currently in use

Quantum

Approximate number of KB of memory in a memory quantum

The next group is **Scans**:

Total

The total number of scan threads as specified by the DS_MAX_SCANS configuration parameter

Free

Number of scan threads currently available for decision-support queries

Quantum

The number of scan threads in a scan-thread quantum

The last group in this portion of the output describes MGM **Load Control**:

Memory

Number of queries that are waiting for memory

Scans

Number of queries that are waiting for scans

Priority

Number of queries that are waiting for queries with higher PDQ priority to run

Max Queries

Number of queries that are waiting for a query slot

Reinit

Number of queries that are waiting for running queries to complete after an onmode -M or -Q command

The next portion of the output, **Active Queries**, describes the MGM active and ready queues. This portion of the output shows the number of queries that are waiting at each gate:

Session

The session ID for the session that initiated the query

Query

Address of the internal control block that is associated with the query

Priority

PDQ priority that is assigned to the query

Thread

Thread that registered the query with MGM

Memory

Memory that is currently granted to the query or memory that is reserved for the query (Unit is MGM pages, which is 8 KB.)

Scans

Number of scan threads currently used by the query or number of scan threads that are allocated to the query

Gate

Gate number at which query is waiting

The next portion of the output, **Free Resource**, provides statistics for MGM free resources. The numbers in this portion and in the final portion reflect statistics since system initialization or the last onmode -Q, -M, or -S command. This portion of the output contains the following information:

Average

Average amount of memory and number of scans

Minimum

Minimum available memory and number of scans

The next portion of the output, **Queries**, provides statistics about MGM queries:

Average

Average active and ready queue length

Maximum

Maximum active and ready queue length

Total

Total active and ready queue length

Resource/Lock Cycle Prevention count

Number of times the system immediately activated a query to avoid a potential deadlock. (The database server can detect when some of the queries in its queue might create a deadlock situation if the queries are not run immediately.)

onstat -g nbm command: Print a block bit map

Use the onstat -g nbm command to display the block bit map for the nonresident segments.

Each bit of the bitmap represents a 4 KB block. If the block is used, then the bit is set to 1. If the block is free, the bit is set to 0. The bitmap is shown as a series of hexadecimal numbers. The bits, and therefore the blocks, are numbered starting at 0 so the first block is block 0, the second is block 1, and so on.

Syntax:

```
onstat -gnbm
```

Example output

This example shows the bitmap for the segment of virtual memory at 0x10CC00000. The bitmap itself is at 0x10CC00290. All 1792 blocks of the segment are free except for block 0 and block 1023.

Figure 156. onstat -g nbm command output

```
Block bitmap for virtual segment address 0x10cc00000:
 address = 0x10cc00290, size(bits) = 1792
 used = 1, largest_free = -1
      0:8000000000000000 0000000000000000 0000000000000000 0000000000000000
    256:0000000000000000 0000000000000000 0000000000000000 0000000000000000
    512:0000000000000000 0000000000000000 0000000000000000 0000000000000000
    768:0000000000000000 0000000000000000 0000000000000000 0000000000000001
   1024:0000000000000000 0000000000000000 0000000000000000 0000000000000000
   1280:0000000000000000 0000000000000000 0000000000000000 0000000000000000
   1536:0000000000000000 0000000000000000 0000000000000000 0000000000000000
```

Output description

address

The starting address of the bitmap.

size

The number of bits in the bitmap. This is also the number of 4 KB blocks in the memory segment.

used

The total number of bits in the bitmap that are set to 1. This is also the number of 4 KB blocks that are in use in the memory segment.

largest free

If this is a value other than -1 it is the largest number of consecutive bits that are free, which is also the number of 4 KB blocks in the largest contiguous set of blocks in the memory segment.

A value of -1 means that the largest free space has not been calculated. The database server only calculates the largest free space if it tries to allocate a set of blocks starting at the *lastalloc* block but there is not enough free space. The value is set to -1 again as soon as another block is allocated in the segment.

onstat -g nsc command: Print current shared memory connection information

Use the onstat -g nsc command to display information about shared memory connections either for all of the current connections or for a specified connection ID.

Syntax:

onstat -gnsc [*client_id*]

If no *client_id* is provided, information about all current shared memory connections to the database server is given. If a *client_id* is provided then this command gives more detailed information about the shared memory connection with that ID.

Example output

This is output of onstat -g nsc with no *client_id*. It shows that there is only one user currently connecting to the database server through shared memory. That connection has an ID of 0.

Figure 158. onstat -g nsc command output

```
  clientid  clientPID      state #serverbufs #clientbufs    #rdwrts
         0       6031  Connected           4           4         12
```

This example shows output from running the command using a *client_id* of 0.

Figure 159. onstat -g nsc command with client id output

```
Network Shared Memory Status for Client: 0

    clientid    clientPID        state  #serverbufs  #clientbufs      #rdwrts
           0        18949    Connected            4            4       447048

     needbuf        segid        semid       semnum     be_semid    be_semnum
           0         1303       851969            0       851969           10

  be_curread  be_curwrite   fe_curread  fe_curwrite
          -1            1            0            2

 be_nextread be_nextwrite  fe_nextread fe_nextwrite
           2            2            4            3

readyqueue
  -1   -1   -1   -1   -1   -1   -1   -1   -1   -1   -1

     Server Buffers                           Client Buffers
  i: bufid     status   offset  fe_addr         bufid     status   offset  fe_addr
  0:     4      inuse     4474   804474             0      avail     3424   803424
  1:     5      inuse     4888   804888             1      avail     3838   803838
  2:     6      avail     4c9c   804c9c             2      inuse     3c4c   803c4c
  3:     7      avail     50b0   8050b0             3      avail     4060   804060
  4:    -1       free        0        0            -1       free        0        0
  5:    -1       free        0        0            -1       free        0        0
```

Output description

clientid

Server assigned ID

clientPID

Client process ID

state

State of connection

Connected

The client has established a connection with the server.

Con1

The server has successfully set up a connection with the client, but the client has not yet been notified of it.

Waiting

The server is in the process of setting up a connection with the client.

Reject

Client connection has been rejected by the server, normally because the server is shutting down or not yet in on-line mode.

Closed

Server has closed the connection with the client. Client might not be aware of the fact yet.

Not connected

Server is initializing internal structures for the connection.

Unknown

Connection has been closed and the client is aware of the fact. Server is cleaning up internal structures.

#serverbufs

Database server buffers currently allocated

#clientbufs

Client buffers currently allocated

#rdwrts

The total number of reads and writes performed through this connection since it was created.

The following items are only in the output if you run the onstat -g nsc command with a *client_id*:

needbuf

Indicates if server is waiting for a buffer to be freed

0

False

1

True

segid

Shared memory segment ID

semid

Semaphore ID

semnum

Semaphore number in the semaphore ID

be_semid

Backend semaphore ID

be_semnum

Backend semaphore number in the semaphore ID

be_curread

ID of backend buffer being read

be_curwrite

ID of backend buffer being written

fe_curread

ID of frontend buffer being read

fe_currwrite

ID of frontend buffer being written

be_nextread

ID of next backend buffer to be read

be_nextwrite

ID of next backend buffer to be written

fe_nextread

ID of next frontend buffer to be read

fe_nextwrite

ID of next frontend buffer to be written

readyqueue

Queue of the shared memory buffer ids

Buffers

i

Internal location key of message buffer

bufid

Message buffer ID

status

Status of message buffer

offset

Offset of memory buffer in shared memory segments

fe_addr

Frontend address of message buffer

onstat -g nsd command: Print poll threads shared-memory data

Use the onstat -g nsd command to display information about shared-memory data for poll threads.

Syntax:

`onstat -gnsd`

Example output

Figure 161. onstat -g nsd command output

```
Network Shared Memory Data for Poll Thread: 0
Free Message Buffer Bitmap
(bitmap address = 10b9eef80, bitmap size 480)
000000010b9eef80:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
000000010b9eefa0:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
Free Message Buffer Status Bitmap
(bitmap address = 10ca0a9b0, bitmap size 50)
000000010ca0a9b0:ffffffff ffffff
Message Buffer Table
bufid   clientid             addr
Message Buffer Status Table
clientid      netscb addr             addr           offset
```

onstat -g nss command: Print shared memory network connections status

Use the onstat -g nss *sessionid* command to display information about the status of the shared memory network connections.

Syntax:

onstat -gnss [*sessionid*]

If no *sessionid* is provided, a one-line summary for each shared memory connection is listed.

Example output

Figure 163. onstat -g nss command output

```
 clientid  clientPID   state    #serverbufs #clientbufs    #rdwrts
        1      14018  Connected        4           4        331
        0      12398  Connected        4           4        294
        2      14036  Connected        4           4         59
```

Output description

clientid (decimal)

Server assigned value for lookups

clientPID (decimal)

Client process ID

state (string)

Current® state of the connection.

- Connected
- Con1
- Waiting
- Reject
- Bedcover
- Closed
- Not connected
- Unknown

#serverbufs (dec)

Number of database server buffers currently allocated

#clientbufs (dec)

Number of client buffers currently allocated

#rdwrts (dec)

Total number of buffers in use

onstat -g ntd command: Print network statistics

Use the onstat -g ntd command to display network statistics by service.

Syntax:

```
onstat -gntd
```

Example output

Figure 165. onstat -g ntd command output

```
global network information:
  #netscb connects     read    write  q-limits  q-exceed alloc/max
   4/   5       11        0     3546 3549/  10   10/    0    0/    0

Client Type     Calls   Accepted   Rejected       Read       Write
sqlexec         yes           11          0       3531        3540
srvinfx         yes            0          0          0           0
onspace         yes            0          0          4           9
onlog           yes            0          0          0           0
onparam         yes            0          0          0           0
oncheck         yes            0          0          0           0
onmonitor       yes            0          0          0           0
dr_accept       yes            0          0          0           0
cdraccept       no             0          0          0           0
srvstat         yes            0          0          0           0
asfecho         yes            0          0          0           0
listener        yes            0          0         11           0
crsamexec       yes            0          0          0           0
onutil          yes            0          0          0           0
drdaexec        yes            0          0          0           0
smx             yes            0          0          0           0
safe            yes            0          0          0           0
Totals                        11          0       3546        3549
```

onstat -g ntm command: Print network mail statistics

Use the onstat -g ntm command to display statistics about network mail.

Syntax:

```
onstat -gntm
```

Example output

Figure 167. onstat -g ntm command output

```
global network information:
  #netscb connects      read     write  q-limits  q-exceed alloc/max
   4/   5        11         0      3546 3549/   10    10/    0     0/    0

Network mailbox information:
 box    netscb thread name      max received    in box    max in box full signal
   5  f07e8b0 soctcppoll       10       24         0         1         0    yes
   6  f0b6ad8 soctcplst        10        0         0         0         0     no
   7  f0e8b18 soctcplst        10        0         0         0         0     no
```

onstat -g ntt command: Print network user times

Use the onstat -g ntt command to display information about network user times.

Syntax:

```
onstat  -gntt
```

Example output

Figure 169. onstat -g ntt command output

```
global network information:
  #netscb connects      read     write     q-limits  q-exceed alloc/max
   3/   3         0         0         0     135/   10     0/    0     2/    0
Individual thread network information (times):
 netscb   thread name    sid  open      read     write      address
c76ea28   ontape         61   14:34:48 14:34:50 14:34:50
c63e548   tlitcplst       4   14:30:43 14:34:48           server.ibm.com|5006|tlitcp
c631028   tlitcppoll      3   14:32:32
```

onstat -g ntu command: Print network user statistics

Use the onstat -g ntu command to display information about network user statistics.

Syntax:

```
onstat  -gntu
```

Example output

Figure 171. onstat -g ntu command output

```
global network information:
  #netscb connects     read    write    q-free  q-limits  q-exceed alloc/max
   2/   3       16     2611     2603    1/    1  135/  10    0/    0    1/    1

Individual thread network information (basic):
  netscb type   thread name    sid   fd poll    reads   writes q-nrm q-pvt q-exp
 d1769f0 soctcp soctcplst        3    1    5       16        0  0/ 0  0/ 0  0/ 0
 d1199f0 soctcp soctcppoll       2    0    5     2595        0  0/ 0  0/ 0  0/ 0
```

onstat -g opn command: Print open partitions

Use the onstat -g opn command to display a list of the partitions (tables and indexes), by thread ID, that are currently open in the system.

Use the ***thread_id*** option to restrict the list to a specified ID.

Syntax:

onstat -gopn [*thread_id*]

Output description

This information is used by HCL Software Support. The output might change over time and depends on your product version or fix pack.

onstat -g osi: Print operating system information

Use the onstat -g osi command to display information on your operating system resources and parameters, including shared memory and semaphore parameters, the amount of memory currently configured on the computer, and the amount of memory that is unused.

The onstat -g osi command also displays statistics on the hardware processors on your computer.

Use this command when the server is not online.

Example Output

Figure 173. onstat -g osi Command Output

```
Machine Configuration....
OS Name                              Linux
OS Release                           2.6.9-34.ELsmp
OS Node Name                         idas
OS Version                           #1 SMP
OS Machine                           x86_64
Number of processors                 4
Number of online processors          4
System memory page size              4096 bytes
System memory                        7970 MB
System free memory                   1536 MB
Number of open files per process     1024
shmmax              33554432
shmmin              1
shmids              4096
shmNumSegs          2097152
semmap              << UnSupported >>
semids              128
semnum              32000
semundo             << UnSupported >>
semNumPerID         250
semops              32
semUndoPerProc      << UnSupported >>
semUndoSize         20
semMaxValue         32767
```

onstat -g pd command: Print push data session-related information

Use the onstat -g pd command to display information about the push data session.

Syntax:

onstat -gpd [*session_id*]

You can specify one of the following invocations.

onstat -g pd

Displays a one-line summary for each session

onstat -g pd *session_id*

Displays information for a specific session

Example output for all sessions

Figure 175. onstat -g pd command output

```
push-data subsystem structure at 0x4eebb028
push-data session structure at 0x4eecc028
push-data sql session id: 0 0x0
Marked as detachable session, session unique id: 2
Smartblob file descriptor: 39
Number of event conditions: 0
Number of pending event operations: 0
Number of discarded event operations: 0
Total event operations returned to client:
```

Example output for a specific session

Figure 176. onstat -g pd *98* command output

```
    push-data subsystem structure at 0x4eebb028
push-data session structure at 0x4eecc028
push-data sql session id: 98 0x62
Marked as detachable session, session unique id: 2
Smartblob file descriptor: 39
Number of event conditions: 1
Number of pending event operations: 0
Number of discarded event operations: 0
Total event operations returned to client: 0
```

onstat -g pd event command: Print push data event-related information

Use the onstat -g pd event command to display information about the push data event.

Syntax:

`onstat -gpd` [*session_id*] `event`

You can specify one of the following invocations.

onstat -g pd event

Displays a one-line summary for each event

onstat -g pd *session_id* event

Displays information of an event for a specific session

Example output for all events

Figure 178. onstat -g pd event command output

```
OneDB Version 1.0.1.0 -- On-Line -- Up 00:20:13 -- 185676 Kbytes
push-data subsystem structure at 0x4eebb028
push-data session structure at 0x4eecc028
push-data sql session id: 98 0x62
Marked as detachable session, session unique id: 2
Number of event conditions: 1
    Push-data event structure at 0x4ece0028
        Full Table Name: test:informix.t1
User data:
Replicate name: pushrepl_98_1497908205_1628814989
```

Note:

- Events can only be registered on tables with logging enabled.
- Events require a primary key, a unique index, or ER key to register events on a table.
- Events cannot be registered on sysmaster pseudo tables.
- Events cannot be registered on timeseries VTI tables.
- Event condition SELECT statements cannot include large objects such as byte, text, blob, clob, or collection datatypes.
- WHERE clauses of event condition SELECT statements cannot refer to other tables or contain sub-queries.
- A read call always returns completed event documents.
- The following message is returned upon timeout from the read API:

  ```
  {ifx_isTimeout:"true"}
  ```

- The following message is returned if event documents are discarded from exceeding the max_pending_ops attribute threshold. The document contains the cumulative count of the total number of discarded event documents.

  ```
  {ifx_warn_total_skipcount:10}
  ```

- The following error message is returned with the ifx_error attribute if the input buffer size is too small, or when other fatal error conditions arise:

  ```
  {ifx_error:" Smartblob read API buffer size ## is too small, expected
          size should be atleast ##"}
  ```

- Event data will not be staged if push-data client is disconnected from the server.
- The client cannot read events for the past time. The commit_logid, commit_logpos and commit_time values in the input document cannot be set for the past time.
- The smartblob read API always returns data in JSON document format. This includes event data, warning messages, and error conditions.
- The input buffer value that is passed to the smartblob read API should be at least 1KB in size.
- When the maxrecs attribute for the session is set to more than one record, then the smartblob read API can return data for multiple events in one read call. The format of the output document format is as follows:

  ```
  { [ {document1}, {document2}]}
  ```

- When the server is restarted, the push-data client might receive duplicates of event data. Therefore, it is recommended to discard duplicate event data by saving the last read event commit_logid, and commit_logpos records, and use this commit log position to register push-data event conditions with the server.
- While registering new push-data event conditions, an internal transient cascade replicate definition is created. The replicate definition gets deleted when a session disconnects from the server. Cascading logic are added to capture changes applied by ER apply threads.
- Detached sessions will be marked as detachable session, session unique id: 2

onstat -g pos command: Print file values

Use the onstat -g pos command to display the values in the **$ONEDB_HOME/etc/.infos.DBSERVERNAME** file.

Syntax:

onstat -gpos

Example output

Figure 180. onstat -g pos command output

```
1   7   0 infos ver/size 3 264
2   1   0 snum     0 52564801 44000000 4139 demo_on
3   4   0 onconfig path /opt/IBM/informix/etc/onconfig.demo_on
4   5   0 host informixva
5   6   0 oninit ver IBM Informix Dynamic Server Version 11.70.UC2DE
6   8   0 sqlhosts path /data/IBM/informix/etc/sqlhosts.demos
7   3 -32767 sema 32769
8   2 -32768 shm  32768 52564801 44000000 114176000 R
9   2   1 shm      1 52564802 4ace3000 67108864 V
```

onstat -g ppd command: Print partition compression dictionary information

Use the onstat –g ppd command to display information about the active compression dictionaries that were created for compressed tables and table fragments or compressed B-tree indexes. You can choose to print information for a particular numbered partition or for all open partitions.

The onstat –g ppd command prints the same information that the **syscompdicts_full** table and the **syscompdicts** view in the **sysmaster** database display. The only difference is that the **syscompdicts_full** table and the **syscompdicts** view display information about all compression dictionaries, not just the active dictionaries.

Syntax:

onstat -gppd { [*partition number*] **| 0** }

If you specify a partition number, onstat -g ppd prints the partition profile for that partition. If you specify `0`, this option prints profiles for all partitions.

Example output

Figure 182. onstat –g ppd Output

```
partnum    Version  DbsNum  CrTS         CrLogID  CrLogPos  DrTS   DrLogID  DrLogPos
0x200002   1        2       1229018150   3        577560    0      0        0
0x200003   1        2       1229018150   3        606232    0      0        0
0x300002   1        3       1229018150   3        630808    0      0        0
0x400002   1        4       1229018150   3        655384    0      0        0
0x500002   1        5       1229018150   3        679960    0      0        0
```

```
partnum    ColOffset  DbsNum    CrTS        CrLogID    CrLogPos  DrTS     DrLogID  DrLogPos
0x1001d5   -1         1         1393371661  4          16339024  0        0        0
0x1001d5   4          1         1393371661  4          16355408  0        0        0
```

Output description

partnum

Partition number to which the compression dictionary applies

Version

Version of the code that is creating the compression dictionary

ColOffset

The byte offset for a compressed partition blob column. `-1` means that only the row is compressed

DbsNum

Number of the dbspace that the dictionary resides in

CrTS

Timestamp that shows when the dictionary was created

CrLogID

Unique ID for the logical log that was created when the dictionary was created

CrLogPos

Position within the logical log when the dictionary was created

DrTS

Timestamp that shows when the dictionary was purged

DrLogID

Unique ID for the logical log that was created when the dictionary was purged

DrLogPos

Position within the logical log when the dictionary was purged

onstat -g ppf command: Print partition profiles

Use the onstat -g ppf ***partition_number*** command to display the partition profile for the specified partition number.

Use the onstat -g ppf or the onstat -g ppf 0 command to display the profiles for all partitions. If the TBLSPACE_STATS configuration parameter is set to 0, then the onstat -g ppf command displays: `Partition profiles disabled`.

For more information on the onstat -g ppf command, see the *HCL OneDB™ Performance Guide*.

Syntax:

onstat -gppf { *partition_number* | **0** }

Example output

Figure 184. onstat -g ppf command output

```
Partition profiles
partnum    lkrqs lkwts dlks  touts isrd  iswrt isrwt isdel bfrd  bfwrt seqsc rhitratio
0x100001   0     0     0     0     0     0     0     0     0     0     0     0
0x100002   1506  0     0     0     416   4     0     4     1282  20    0     97
0x100003   15    0     0     0     5     0     0     0     20    0     0     75
0x1000a5   0     0     0     0     0     0     0     0     12    0     0     67
0x1000e3   4     0     0     0     1     0     0     0     4     0     0     25
0x200001   0     0     0     0     0     0     0     0     0     0     0     0
0x300001   0     0     0     0     0     0     0     0     0     0     0     0
0x400001   0     0     0     0     0     0     0     0     0     0     0     0
```

Output description

partnum (hex)

The partition number

lkrqs (decimal)

The number of lock requests for a partition

lkwts (decimal)

The number of lock waits for a partition

dlks (decimal)

The number of deadlocks for a partition

touts(decimal)

The number of remote deadlock timeouts for a partition

isrd (decimal)

The number of read operations for a partition

iswrt (decimal)

The number of write operations for a partition

isrwt (decimal)

The number of rewrite or update operations for a partition

isdel (decimal)

The number of delete operations for a partition

bfrd (decimal)

The number of buffer read operations, in pages

bfwrt (decimal)

The number of buffer write operations, in pages

seqsc (decimal)

The number of sequential scans for a partition

rhitratio (percentage)

The ratio of disk read operations to buffer read operations

onstat -g pqs command: Print operators for all SQL queries

Use the onstat –g pqs command to display information about the operators used in all of the SQL queries that are currently running.

You can use this command to troubleshoot an application, to find which operators are running for the query and for how long, and how many rows each operator returns. While the EXPLAIN file contains information that will give you a general sense of the query plan, the onstat –g pqs command displays the runtime operator information for the query and the query plan.

Syntax:

`onstat -gpqs` [*sessionid*]

You can specify one of the following invocations:

Table 158. Descriptions of each onstat -g pqs command invocation

Invocation	Explanation
onstat -g pqs	Displays a one-line summary for each session.
onstat -g pqs *sessionid*	Displays information for the session that you specify.

Example output

The following example shows the results when three separate SQL statements are run in different sessions. The statements are:

```
select * from syscolumns;
select * from systables a, systables b;
update t1 set rowsize = rowsize +100;
```

Figure 186. onstat –g pqs command output

```
Query Operators:
addr      ses-id    opname    phase    rows    time           in1         in2     stmt-type
ae50b3a   23        scan      open     0       00:00.00         0           0     SELECT
af269d0    5        nljoin    next     224717 00:01.82    af26a90     aeb4478     SELECT
af26a90    5        scan      next     472     00:00.20         0           0     SELECT
aeb4478    5        scan      next     50      00:01.63         0           0     SELECT
ad3c530   26        scan      open     0       00:00.00         0           0     UPDATE (all)
```

Output description

addr

The address of the operator in memory. You can use this address to track which SCAN operator belongs to each JOIN operator.

ses-id

The session ID in which the SQL statement was run.

opname

The name of the operator.

phase

The phase in which the operator was used. For example OPEN, NEXT, CLOSE.

rows

The number of rows that are processed by the operator.

time

The amount of time to process the operator. The time is displayed to the millisecond. A time of 01:20.10 is 1 minute, 20 seconds, and 10 milliseconds.

in1

The first (outer) operator in the join.

in2

The second (inner) operator in the join.

stmt-type

The type of SQL statement, such as SELECT, UPDATE, DELETE.

onstat -g prc command: Print sessions using UDR or SPL routines

Use the onstat -g prc command to display the number of sessions that are currently using the UDR or SPL routine.

Syntax:

```
onstat  -gprc
```

Example output

Figure 188. onstat -g prc command output

```
UDR Cache:
    Number of lists          : 31
    PC_POOLSIZE              : 127

UDR Cache Entries:

list id   ref  drop hits       last_access          heap_ptr         udr_name
--------------------------------------------------------------------------------


0    79   0    0    1          2020-05-12 10:36:34  464a1c38         stores_demo@myserver:.proc1
0    46   0    0    10         2020-05-12 10:36:34  4dcba038         stores_demo@myserver:.proc2
0    376  0    0    1          2020-05-12 10:36:32  46303038         stores_demo@myserver:.proc3
0    363  0    0    1          2020-05-12 10:36:32  462f2c38         stores_demo@myserver:.proc4

    Total number of udr entries : 254
    Number of entries in use    : 9
```

Output description

Number of lists

Number of lists in the UDR cache

PC_POOLSIZE

Number of entries that can be cached at one time

Number of entries

Number of entries in the UDR cache

Number of inuse entries

Number of entries that are being used

list

list#

UDR cache hash chain ID (bucket number)

id

Unique ID of the routine

ref

ref_cnt

Number of sessions that are currently accessing the UDR or SPL routine from the cache

drop

dropped?

Whether the routine is marked to be dropped

hits

The number of times the cache entry is accessed.

last_access

The time at which the cache entry was last accessed.

heap_ptr

Heap address that is used to store this entry

udr_name

The name of the UDR or SPL routine in the cache

Total number of udr entries

Number of entries in the cache

Number of entries in use

Number of entries that are being used

onstat -g proxy command: Print proxy distributor information

Use the onstat -g proxy command to display information about proxy distributors. The output of the onstat -g proxy command differs slightly depending on whether the command is run on a primary server or on a secondary server.

Syntax:

onstat -gproxy { **all** | [*proxy_id* [*proxy_transaction_id* [{ *sequence_number* }]]] }

Invocation	Explanation
onstat -g proxy	Displays proxy distributor information
onstat -g proxy all	When run on the primary server, displays information about proxy distributors and proxy agent threads. When run on the secondary server, displays information about all sessions currently performing updates to secondary servers.
onstat -g proxy ***proxy_id proxy_transaction_id sequence_number***	This option is valid only on secondary servers. Displays detailed information about the current work bring performed by a given proxy distributor. The *proxy_transaction_id* and *sequence_number* are optional parameters. When supplied, the first number is considered the *proxy_transaction_id*, and the second is interpreted as the *sequence_number*. If the supplied *proxy_transaction_id* or *sequence_number* do not exist, the command output is the same as the output for onstat -

Example output using the `onstat -g proxy` command on a primary server

Figure 190. onstat -g proxy command output (run from primary server)

```
Secondary          Proxy      Reference Transaction  Hot Row
Node               ID         Count    Count        Total
nagpur_sdc1        2619       0        2            0
nagpur_c2          2632       0        1            0
nagpur_sec         2633       0        1            0   I
```

Output description

Secondary Node

Name of the secondary server as it is known by the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor.

Example output using the onstat -g proxy command on a secondary server

Figure 191. onstat -g proxy command output (run from secondary server)

```
Primary            Proxy      Reference Transaction  Hot Row
Node               ID         Count     Count        Total
nagpur             2619       5         2            0
```

Output description

Primary Node

Name of the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor.

Example output using the onstat -g proxy all command on a primary server

Figure 192. onstat -g proxy all command output (run from primary server)

```
Secondary          Proxy      Reference Transaction  Hot Row
Node               ID         Count     Count        Total
nagpur_sdc1        2619       0         2            0
nagpur_c2          2632       0         1            0
nagpur_sec         2633       0         1            0

TID      Flags      Proxy  Source   Proxy    Current  sqlerrno iserrno
                    ID     SessID   TxnID    Seq
94       0x00000224 2619   21       1        29        0        0
95       0x00000224 2619   22       2        68        0        0
93       0x00000224 2632   21       2        2         0        0
91       0x00000224 2633   25       1        6         0        0
```

Output description

Secondary Node

Name of the secondary server as it is known by the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor.

TID

ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the session on the secondary server.

Flags

Flags of the proxy agent thread.

Proxy ID

The ID of the proxy distributor on behalf of which the proxy agent thread (TID) is running.

Source SessID

The ID of the user's session on the secondary server.

Proxy TxnID

The number of the current transaction. These numbers are unique to the proxy distributor.

Current® Seq

The sequence number of the current operation in the current transaction.

sqlerrno

The error number of any SQL error (or 0 if no errors).

iserrno

The error number of any ISAM or RSAM error (or 0 if no errors).

Example output using the onstat -g proxy all command on a secondary server

Figure 193. onstat -g proxy all command output (run from secondary server)

```
Primary            Proxy      Reference Transaction  Hot Row
Node               ID         Count     Count        Total
nagpur             2619       5         2            0

Session  Session  Proxy    Proxy    Proxy    Current  Pending  Reference
         Ref      Proxy_id TID      TxnID    Seq      Ops      Count
21       2        2619     94       1        29       1        1
22       2        2619     95       2        68       1        1
```

Output description

Primary Node

Name of the primary server.

Proxy ID

ID of the proxy distributor. Proxy IDs are unique within a high-availability cluster.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Transaction Count

The number of transactions currently being processed by the proxy distributor.

Hot Row Total

Total number of hot rows ever handled by the proxy distributor. A hot row is a row on a secondary server that is updated multiple times by more than one client. When a row is updated multiple times, the secondary server reads the before image from the primary server by placing an update lock on the row if the most recent update operation from a different session is not replayed on the secondary server.

Session

The session ID

Proxy ID

The ID of the proxy distributor on behalf of which the proxy agent thread (TID) is running.

Proxy TID

Transaction ID of the proxy agent thread running on the primary server. This ID is created by the proxy distributor to handle work from the secondary server session.

Proxy TxnID

The number of the current transaction. These numbers are unique to the proxy distributor.

Current® Seq

The sequence number of the current operation in the current transaction.

Pending Ops

The number of operations buffered on the secondary server that have not yet been sent to the primary server.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Example output using the *proxy_id* option on a secondary server

This command returns information only on a secondary server.

Figure 194. onstat -g proxy proxy_id command output (run from secondary server)

```
Proxy    Reference Pending  ProxySID
TxnID    Count     Ops
1        1         1        3
2        1         1        4
```

Output description

Proxy TxnID

The number of the current transaction. These numbers are unique to the proxy distributor.

Reference Count

Indicates the number of threads that are using the information for the current transaction. When the count becomes 0, the transaction processing is complete (either successfully or unsuccessfully).

Pending Ops

The number of operations buffered on the secondary server that have not yet been sent to the primary server.

Proxy SID

Proxy session ID.

Example output using the *proxy_id proxy_transaction_id* options on a secondary server

This command returns information only on a secondary server.

Figure 195. onstat -g proxy_id proxy_transaction_id command output (run from secondary server)

```
Sequence Operation rowid    Table                          sqlerrno
Number   Type               Name
28       *Update   526      stores_demo:nilesho.customer   0
```

Output description

Sequence Number

The number of the operation.

Operation Type

The type of operation to be performed. One of: Insert, Update, Delete, Other.

rowid

The row ID of the row in which to apply the operation.

Table Name

The full table name, trimmed to fit a reasonable length. Format: database.owner.tablename

sqlerrno

The error number of any SQL error (or 0 if no errors).

Example output using the *proxy_id proxy_transaction_id sequence_number* options on a secondary server

This command returns information only on a secondary server.

The output fields are the same as the output fields displayed for the onstat -g proxy_id proxy_transaction_id command. While the onstat -g proxy_id proxy_transaction_id command displays details for a transaction, the onstat -g proxy_id proxy_transaction_id sequence_number displays details for all transaction operations.

Figure 196. onstat -g proxy_id proxy_transaction_id sequence_number command output (run from secondary server)

```
s
Proxy    Reference Pending  ProxySID
TxnID    Count     Ops
61       0         3        22

onstat -g proxy 2788 61


Sequence Operation rowid    Table                          sqlerrno
Number   Type               Name
960      Update    264      stores_demo:nilesho.customer      0
961      Update    265      stores_demo:nilesho.orders        0
962      Update    266      stores_demo:nilesho.items         0

onstat -g proxy 2788 61 962


Sequence Operation rowid    Table                          sqlerrno
Number   Type               Name
962      Update    266      stores_demo:nilesho.items         0
```

onstat -g qst command: Print wait options for mutex and condition queues

Use the onstat -g qst command to display the wait statistics for mutex queues and condition queues (queues of waiters for a mutex or a condition).

The QSTATS configuration parameter must be set to `1` to enable the collection of statistics. For more information, see QSTATS configuration parameter on page 148.

Syntax:

onstat -gqst

Example output

Figure 198. onstat -g qst command output

```

Mutex Queue Statistics
name     nwaits   avg_time max_time avgq maxq nservs   avg_time

ddh chai 1        1354863  1354863  1    1    56       1690

Condition Queue Statistics
name     nwaits   avg_time max_time avgq maxq nservs   avg_time

arrived  1        110008   110008   1    1    0        0
logbf0   21       642      4431     1    2    0        0
logbf1   15       475      2519     1    2    0        0
logbf2   19       596      3274     1    2    0        0
bp_cond  1        0        0        1    1    0        0
```

Output description

name (string)

Name of the mutex or condition resource being waited for

nwaits (decimal)

Number of times this resource was waited for

avg_time (decimal)

Average time spent waiting (in microseconds)

max_time (decimal)

Maximum time spent waiting (in microseconds)

avgq (decimal)

Average length of the queue

maxq (decimal)

Maximum length of the queue

nservs (decimal)

Number of times this resource was acquired

avg_time (decimal, microsecond)

Average time the resource was held per acquisition (in microseconds)

onstat -g rah command: Print read-ahead request statistics

Use the onstat -g rah command to display information about read-ahead requests.

Syntax:

```
onstat  -grah
```

Example output

Figure 200. onstat -g rah command output

```
Read Ahead

# Qs                  1
# threads             2
# Requests            58690
# Continued           0
# Memory Failures     0
Last Thread Add       04/06/2013.14:34
Way behind            0

Partition ReadAhead Statistics

          Buffer   Disk   Hit    Data       Index        Idx/Dat     Log/PageList Last Committed
Partnum   Reads    Reads  Ratio  # Reqs Eff # Reqs   Eff  # Reqs   Eff # Pages Eff  # Reqs  Eff   # Resch
0x200003  4312677  110    99     0      0   0        0    0        0    0      0     12906  100   0
0x300002  23740584 1427   99     0      0   0        0    0        0    0      0     6681   100   7
0x400002  17818942 966    99     0      0   0        0    0        0    0      0     25849  100   57
```

```
Read Ahead

# threads             4
# Requests            150501
# Continued  1017
# Memory Failures     0
Q depth               0
Last Thread Add       04/06/2011.14:34
Way behind            0

Partition ReadAhead Statistics

          Buffer  Disk    Hit    Data          Index         Idx/Dat
Partnum   Reads   Reads   Ratio  # Reqs   Eff   # Reqs  Eff   # Reqs  Eff
0x200003  2412176 587381  75     0        0     150499  67    0       0
```

Output description

Qs

Number of queues for read-ahead requests

 # threads

Number of read-ahead threads

 # Requests

Number of read-ahead requests

 # Continued

Number of times a read-ahead request continued to occur

 # Memory Failures

Number of failed requests because of insufficient memory

Q Depth

Depth of request queue

Last Thread Add

Date and time when the last read-ahead thread was added

Way behind

How many page list requests were dropped because the read-ahead daemon is too far behind

Partnum

Partition number

Buffer reads

Number of bufferpool and disk pages that were read

Disk Reads

Number of pages that were read from disk

Hit Ratio

Cache hit ratio for the partition

 # Reqs

Number of read ahead requests. (There are 5 instances of this output field: for data, the index, index data, log pages, and last committed rows.)

Number of read ahead requests. (There are three instances of this output field: for data, the index, and index data.)

Eff

Efficiency of the read-ahead requests. This is the ratio been the number of pages requested by read-ahead operations to the number of pages that were already cached and for which a read-ahead operations was not needed. Values are between 0 and 100. A higher number means that read ahead is beneficial. (There are 5 instances of this output field: for data, the index, index data, log pages, and last committed rows.)

Efficiency of the read-ahead requests. This is the ratio been the number of pages requested by read-ahead operations to the number of pages that were already cached and for which a read-ahead operations was not needed. Values are between 0 and 100. A higher number means that read ahead is beneficial. (There are three instances of this output field: for data, for the index, and for index data.)

Resch

The number of requests for last committed rows that are rescheduled because the updates to a multi-piece row are not complete.

onstat -g rbm command: Print a block map of shared memory

Use the onstat -g rbm command to display a hexadecimal bitmap of the free and used blocks within the resident segment of shared memory.

Syntax:

```
onstat  -grbm
```

Example output

Figure 202. onstat -g rbm command output

```
Block bitmap for resident segment address 0x44000000:
 address = 0x440003bc, size(bits) = 3035
 used = 3031, largest_free = 4

      0:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
    256:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
    512:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
    768:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   1024:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   1280:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   1536:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   1792:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   2048:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   2304:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   2560:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff
   2816:ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffe00
```

Output description

Header

address (hex)

In-memory starting address of the used/free blocks in the segment

size (bits)

Number of bits in the block bitmap; each bit represents one block

used (blocks)

Used blocks in the bitmap

largest_free (blocks)

Largest run of free blocks

Data

Bit number (decimal): data (hex)

Bit number followed by 32 bytes of data (hex)

onstat -g rea command: Print ready threads

Use the onstat -g rea command to display information about the virtual processor threads whose current status is ready.

Syntax:

```
onstat -grea
```

Example output

Following is sample output from the onstat -g rea command. For a description of the output, see onstat -g ath command: Print information about all threads on page 512.

Figure 204. onstat -g rea command output

```
Ready threads:
tid     tcb     rstcb   prty    status   vp-class   name
6       536a38  406464  4       ready       3cpu    main_loop()
28      60cfe8  40a124  4       ready       1cpu    onmode_mon
33      672a20  409dc4  2       ready       3cpu    sqlexec
```

onstat -g rss command: Print RS secondary server information

Use the onstat -g rss commands to display information about remote standalone secondary servers.

Syntax:

onstat -grss [{ verbose | log | *server_name* }]

The output of the onstat -g rss command differs slightly depending on whether the command is run on the primary server or on the RS secondary server.

Invocation	Explanation
onstat -g rss	Displays brief RS secondary server information
onstat -g rss verbose	Displays detailed RS secondary server information
onstat -g rss log	Displays log information. This command is only applicable when run on the primary server.
onstat -g rss ***server_name***	Displays information about a specific RS secondary server. This command is only applicable when run on the primary server.

Example output (primary server)

Figure 206. onstat -g rss verbose command output, when the command is run on the primary server.

```
Local server type: Primary
Index page logging status: Enabled
Index page logging was enabled at: 2020/05/23 06:12:06
Number of RSS servers: 2

RSS Server information:

RSS Server control block: 0x64f64758
RSS server name: rahulb_3
RSS server status: Active
RSS connection status: Connected
RSS flow control:384/352
Log transmission status: Active
Next log page to send(log id,page): 6,36
Last log page acked(log id,page): 6,35
Last log page applied(log id,page): 6,35
Time of Last Acknowledgement: 2020-05-23.06:13:29
Pending Log Pages to be ACKed: 0
Approximate Log Page Backlog:0
Sequence number of next buffer to send: 231
Sequence number of last buffer acked: 230
Supports Proxy Writes: N
Total number of delay(s): 8
Time of last delay: 2020-05-23.06:13:02
```

Output description (primary server)

Local server type

Primary or RSS (remote standalone secondary) server type

Index page logging status

Displays whether index page logging is enabled or disabled between primary server and secondary server

Index page logging was enabled at

Date and time that index page logging was enabled

Number of RSS servers

Number of RS secondary servers connected to the primary server

RSS Server control block

RS secondary server control block

RSS Server name

Name of RS secondary server

RSS Server status

Displays whether RS secondary server is active or not

RSS flow control

Values, in number of logical log pages, determining when flow control is enabled or disabled, respectively.

RSS Connection status

Connection status of RS secondary server

Log transmission status

Displays whether log transmission is active or inactive

Next log page to send (log id, page)

The log ID and page number of the next log page that will be sent

Last log page acked (log id, page)

The log ID and page number of the last acknowledged log

Last log page applied (log id, page)

The log ID and page number of the last applied log

Time of Last Acknowledgment

The time at which the last log was acknowledged

Pending Log pages to be ACKed

The number of logs sent but not yet acknowledged

Approximate Log Page Backlog

The difference between the number of logs that were sent and the end of the logical log

Sequence number of next buffer to send

The sequence number of the next buffer to be sent

Sequence number of last buffer acked

The sequence number of the last acknowledged buffer

Supports Proxy Writes

Displays whether the server is currently configured to allow updates to secondary servers. **Y** = supports updates to secondary servers, **N** = does not support updates to secondary servers.

Total number of delay(s)

The total number of times the flow delay occurred.

Time of last delay

The time of last delay in flow control.

Example output with log option (primary server)

Figure 207. onstat -g rss log command output, when the command is run on the primary server.

```
Log Pages Snooped:
RSS Srv         From      From      Tossed
name             Cache     Disk     (LBC full)

cdr_ol_nag_1_c1    1368       1331      0
cdr_ol_nag_1_c2    1357       1342      0
cdr_ol_nag_1_c3    1356       1343      0
```

Output description with log option (primary server)

Log Pages Snooped

Statistics for each RS secondary server

RSS Srv name

RS secondary server name

From Cache

From cache number

From Disk

Log from disk

Tossed (LBC full)

Number of log pages that were discarded as a result of the LBC becoming full

Example output (RS secondary server)

Figure 208. onstat -g rss command output, when the command is run on the RS secondary server.

```
Local server type: RSS
Server Status: Active
Source server name: cdr_ol_nag_1
Connection status: Connected
Last log page received(log id,page): 7,877
```

Output description (RS secondary server)

Local server type

Primary or RSS (remote standalone secondary) server type

Server Status

Displays whether RS secondary server is active

Source server name

Name of the primary server

Connection status

Connection status of RS secondary server

Last log page received (log id,page)

Most recent log ID and page received

Example output with verbose option (RS secondary server)

Figure 209. onstat -g rss verbose command output, when the command is run on the RS secondary server.

```
RSS Server control block: 0x45a3fe58
Local server type: RSS
Server Status: Active
Source server name: my_server
Connection status: Connected
Last log page received(log id,page): 10,1364
Sequence number of last buffer received: 489
Sequence number of last buffer acked: 489
Delay Apply: Configured (3)
Stop Apply: Not configured.
Delay or Stop Apply control block: 0x45a40ba8
    Pending pages: 7
    Last page written: (10:1372).
    Next page to read: (10:1366).
    Delay or Stop Apply thread: Running.
```

Output description with verbose option (RS secondary server)

RSS Server control block

The server control block.

Local server type

The local server's type.

Server Status

The status of the RS secondary server.

Source server name

The name of the primary server in the RS secondary server's high-availability cluster.

Connection status

The status of the connection between the RS secondary server and the cluster's primary server.

Last log page received (log id,page)

The log ID and page number of the last log acknowledged by the RS secondary server.

Sequence number of last buffer received

The sequence number of the last buffer that was received by the RS secondary server.

Sequence number of last buffer acked

The sequence number of the last buffer acknowledged by the RS secondary server.

Delay Apply

Whether delay apply is configured or not. The delay value, in seconds, is included in parentheses.

Stop Apply

Whether stop apply is configured or not. The stop value, which is enclosed in parentheses, is either 1 or a Unix time.

Delay or Stop Apply control block

The control block of the delay or the stop apply.

Pending pages

The number of pages that are waiting to be written to the log-staging directory.

Last page written

The log id and page number of the log that was most recently written to the log-staging directory.

Next page to read

The log id and page number of the next log to write to the log-staging directory.

Delay or Stop Apply thread

The status of the delay-apply or stop-apply thread.

onstat -g rwm command: Print read and write mutexes

Use the onstat -g rwm command to display information about read, write, and waiting mutex threads, and to list the addresses of the tickets that these threads have acquired.

Syntax:

```
onstat  -grwm
```

Example output

Figure 211. onstat -g rwm command output

```
MUTEX   NAME     write/read/wait    tcb list
<address> <name>       first mutex
       Writer     ticket = <ticket address>   tcb=<thread address> <thread name>
       Readers    ticket = <ticket address>   tcb=<thread address> <thread name>
       Waiters    ticket = <ticket address>   tcb=<thread address> <thread name>
<address> <name>         second mutex
       Writer     ticket = <ticket address>   tcb=<thread address> <thread name>
       Readers     ticket = <ticket address>  tcb=<thread address> <thread name>
       Waiters     ticket = <ticket address>  tcb=<thread address> <thread name>
....
....
....
<address> <name>      last mutex
       Writer     ticket = <ticket address>   tcb=<thread address> <thread name>
       Readers    ticket = <ticket address>   tcb=<thread address> <thread name>
       Waiters    ticket = <ticket address>   tcb=<thread address> <thread name>
```

Output description

tcb

List of thread addresses

Writer

List of write threads

Readers

List of read threads

Waiters

List of waiting threads

ticket

Address of ticket acquired by the thread

onstat -g sch command: Print VP information

Use the onstat -g sch command to display information about thread migration and the number of semaphore operations, spins, and busy waits for each virtual processor.

Syntax:

```
onstat -gsch
```

Example Output

Figure 213. onstat -g sch command output

```
VP Scheduler Statistics:
 vp    pid       class       semops    busy waits  spins/wait
 1     3284      cpu         23997     0           0
 2     1340      adm         0         0           0
 3     4624      lio         2         0           0
 4     3320      pio         2         0           0
 5     6076      aio         7710      0           0
 6     4580      msc         46        0           0
 7     3428      soc         7         0           0
 8     2308      soc         1         0           0

Thread Migration Statistics:
 vp   pid      class steal-at steal-sc idlvp-at idlvp-sc inl-polls Q-ln
 1    3284     cpu   0        0        0        0        0         0
 2    1340     adm   0        0        0        0        0         0
 3    4624     lio   0        0        0        0        0         0
 4    3320     pio   0        0        0        0        0         0
 5    6076     aio   0        0        0        0        0         0
 6    4580     msc   0        0        0        0        0         0
 7    3428     soc   0        0        0        0        0         0
 8    2308     soc   0        0        0        0        0         0
```

onstat -g scn command: Print scan information

Use the onstat -g scn command to display the status of a current scan and information about the scan.

If you have a long-running scan, you might want to use this command to check the progress of the scan, to determine how long the scan will take before it completes, and to view information about the scan. For tables, the onstat -g scn command output identifies whether a scan is a light or bufferpool scan.

Syntax:

onstat -g scn

Example Output

Figure 215. onstat -g scn output showing table information

```
Light Scan Info
descriptor  address           next_lpage  next_ppage        ppage_left  bufcnt  look_aside

RSAM batch sequential scan info

SesID Thread Partnum  Rowid    Rows Scan'd Scan Type Lock Mode  Notes
48    68     10016e   12bb09   43146       Light     Table      Look aside,
40    47     100106   101      0           Buffpool  +Test      Must copy
```

Information about an index scan is valid when a scan is running.

Figure 216. onstat -g scn output showing index scan information

```
RSAM batch index scan info

SesID Thread Partnum  Scan Type Lock Mode  Notes
136   156    100197             SLock+Test
       Start Key   GT   :-2147483648:
       Stop Key    EQ   :1500:
       Current key      :170:
       Current position: buffp 0x10a4bc0c8 pagenum 2 slot 17 rowoff 4 flags 0
```

Output Description

descriptor (decimal)

Light scan ID

address (hex)

Memory address of the light scan descriptor

next_lpage (hex)

Next logical page address to scan

next_ppage (hex)

Next physical page address to scan

ppage_left (decimal)

Number of physical pages left to scan in the current extent

bufcnt

Number of light scan buffers used for this light scan

look_aside

Whether look aside is needed for this light scan (`Y` = yes, `N` = no). Look asides occur when a thread needs to examine the buffer pool for existing pages to obtain the latest image of a page being light scanned.

SesID

Session ID

Thread

Thread ID

Partnum

Partition number

Rowid

Current® row ID

Rows Scan'd

Number of rows that have been scanned

Scan Type

For tables, either:

- `Bufferpool`
- `Light` (light scan)

For indexes, either:

- `key only`
- No value if the scan is not a key-only scan

Lock Mode

The type of acquired lock or no lock:

- `Table` (table-level lock acquired)
- `Slock` (share locks acquired)
- `Ulock` (update locks acquired)
- blank (no locks acquired)

This column can also show one of the following values:

- `+Test` (The scan tested for a conflict with the specified lock type; the lock was not acquired.)
- `+Keep` (The acquired locks will be held until end of session instead of the end of the transaction.)

Notes

This column can show one of the following values:

- `Look aside`

  The light scan is performing look aside.

  The light scan reads blocks of pages directly from disk into large buffers, rather than getting each page from the buffer manager. In some cases, this process requires the light scan to check the buffer pool for the presence of each data page that it processes from one of its large buffers; this process is called *look aside*. If the page is currently in the buffer pool, the light scan will use that copy instead of the one in the light scan large buffer. If the page is not in the buffer pool, the light scan will use the copy that the light scan read from disk into its large buffer. If the light scan is performing look aside, the performance of the scan is slightly reduced.

In many cases, the light scan can detect that it is impossible for the buffer pool to have a newer version of the page. In these situations, the light scan will not check the buffer pool, and the look aside note will be absent.

- `Forward row lookup`

  The server is performing a light scan on a table that has rows that span pages. The light scan must access and use the buffer pool to get the remainder pieces of any rows that are not completely on the home page.

Start key

Start key of the scan

Stop key

End key of the scan

Current® key

The current key in the scan

Current® position

The current location of the scan in the index, for example, the page, slot, and offset

onstat -g sds command: Print SD secondary server information

Use the onstat -g sds command to display information about shared-disk secondary servers.

Syntax:

onstat -g sds [{ *server_name* | **verbose** }]

The output of the onstat -g sds command differs slightly depending on whether the command is issued on the primary server or on the SD secondary server.

Invocation	Explanation
onstat -g sds	Displays brief SD secondary server information
onstat -g sds verbose	Displays detailed SD secondary server information
onstat -g sds ***server_name***	Displays information about a specific SD secondary server. When *server_name* is specified, the command must be issued from the primary server.

Example output (primary server)

Figure 218. onstat -g sds command output when you run the command from primary server.

```
Local server type: Primary
Number of SDS servers:1

SDS server information

SDS srv      SDS srv      Connection        Last LPG sent  Supports
name         status       status            (log id,page)  Proxy Writes
C_151162     Active       Connected             554,4998   Y
```

Output description (primary server)

Local server type

Primary or SDS (shared disk secondary) server type

Number of SDS servers

Number of SD secondary servers connected to the primary server

SDS Srv name

Name of SD secondary server

SDS Srv status

Displays whether SD secondary server is active

Connection status

Displays whether SD secondary server is connected

Last LPG sent (log id, page)

Most recent LPG log ID and page

Supports Proxy Writes

Displays whether the server is currently configured to allow updates to secondary servers. **Y** = supports updates to secondary servers, **N** = does not support updates to secondary servers.

Example output with verbose option (primary server)

Figure 219. onstat -g sds ***server_name*** command output when you run the command from primary server.

```
Number of SDS servers:2
Updater node alias name :server_1

SDS server control block: 0x46217640
server name: rahulb_4
server type: SDS
server status: Active
connection status: Connected
Last log page sent(log id,page):6,44
Last log page flushed(log id,page):6,44
Last log page acked (log id, page):6,44
Last LSN acked (log id,pos):6,180664
Last log page applied(log id,page): 6,44
Approximate Log Page Backlog:0
Current SDS Cycle:19
Acked SDS Cycle:19
Sequence number of next buffer to send: 447
Sequence number of last buffer acked: 444
Time of last ack:2020/05/23 06:16:28
Supports Proxy Writes: N
Time of last received message: 2020/05/23 06:16:49
Time of last alternate write: N/A
Time of last alternate read : N/A
Total number of delay(s): 11
Time of last delay: 2020/05/23 06:13:04
```

Output description with verbose option (primary server)

Number of SDS servers

The number of SD secondary servers that share disk space with the primary server

Updater node alias name

The name of the primary server

SDS server control block

SD secondary server control block

server name

The name of the server

server type

The type of server

server status

Displays whether the server is active or inactive

connection status

Status of connection between primary and secondary server

Last log page sent (log id, page)

Log ID and page of most recent log page sent

Last log page flushed (log id, page)

Log ID and page of the most recent log page flushed

Last log page acked (log id, pos))

Most recent log page acknowledged

Last LSN acked (log id, pos)

Most recent log sequence number that was acknowledged

Last log page applied(log id,page)

The log ID and page number of the last applied log

Approximate Log Page Backlog

The number of logs waiting to be sent

Current® SDS Cycle

Used internally by the support to monitor coordination of the primary server with the SDS server

Acked SDS Cycle

Used internally by the support to monitor coordination of the primary server with the SDS server

Sequence number of next buffer to send

Sequence number of next buffer to send

Sequence number of last buffer acked

Sequence number of next buffer acknowledged

Time of last ack

Date and time of last log acknowledgment

Supports Proxy Writes

Displays whether the server is currently configured to allow updates to secondary servers. **Y** = supports updates to secondary servers, **N** = does not support updates to secondary servers.

Time of last received message:

The timestamp of the current server's most recently received from another server.

Time of last alternate write

The timestamp of the current server's most recent write to the blobspace specified by the SDS_ALTERNATE configuration parameter.

Time of last alternate read

The timestamp of the current server's most recent read from the blobspace specified by the SDS_ALTERNATE configuration parameter.

Total number of delay(s)

The total number of times the flow delay occurred.

Time of last delay

The time of last delay in flow control.

Example output with verbose option (SD secondary server)

Figure 220. onstat -g sds verbose command output when you run the command from the SD secondary server.

```
SDS server control block: 0xb299880
Local server type: SDS
Server Status : Active
Source server name: my_source_server
Connection status: Connected
Last log page received(log id,page): 7,884
Next log page to read(log id,page):7,885
Last LSN acked (log id,pos):7,3621272
Sequence number of last buffer received: 0
Sequence number of last buffer acked: 0
Current paging file:/dbspaces/page_my_source_server_sdc1_
Current paging file size:2048
Old paging file:/dbspaces/page_my_source_server_sdc1_
Old paging file size:10240
```

Output description with verbose option (SD secondary server)

SDS server control block

SD secondary server control block

Local server type

Primary or SDS (shared disk secondary) server type

Server status

Displays whether SD secondary server is active

Source server name

Displays name of primary server

Connection status

Displays whether SD secondary server is connected

Last log page received (log id, page)

Most recent log page received

Next log page to read (log id,page)

Next log page in sequence to read

Last LSN acked (log id,pos)

Most recent LSN acknowledged

Sequence number of last buffer received

Sequence number of last buffer received

Sequence number of last buffer acked

Sequence number of last buffer acknowledged

Current® paging file

Name of current paging file

Current® paging file size

Size of current paging file

Old paging file

Name of previous paging file

Old paging file size

Size of previous paging file

onstat -g seg command: Print shared memory segment statistics

Use the onstat -g seg command to show the statistics for shared memory segments.

This command shows how many segments are attached and their sizes. You can run the onstat -g seg command on a dump file that was created without the buffer pool.

Syntax:

```
onstat  -gseg
```

Example output

Figure 222. onstat -g seg command output

```
Segment Summary:
id          key        addr     size       ovhd     class blkused  blkfree
720914    52e44801  44000000 4390912    248812  R    1072     0
753683    52e44802  44430000 131072000  769136  V    22573    9427
819221    52e44803  4c130000 66027520  1        B    16120    0
851990    52e44804  50028000 83648512  1        B    20422    0
Total:    -          -       285138944 -        -    60187    9427
Virtual segment low memory reserve (bytes):4194304
Low memory reserve used 0 times and used maximum block size 0 bytes
```

```
id          key        addr     size       ovhd     class blkused  blkfree
8945678    52604801   44000000 133652480  1006264  R     32627    3
8978447    52604802   4bf76000 131072000  769136   V     17181    14819
Total:     -          -        264724480  -        -     49808    14822

   (* segment locked in memory)
Virtual segment low memory reserve (bytes):4194304
Low memory reserve used 0 times and used maximum block size 0 bytes
```

Output description

id

The ID of the shared memory segment

key

The shared memory key that is associated with the shared memory segment ID

addr

The address of the shared memory segment

size

The size of the shared memory segment in bytes

ovhd

The size of the shared memory segment control information (overhead) in bytes

class

The class of the shared memory segment (B is for Buffferpool, R is for Resident, V is for Virtual, VX is for Virtual Extended, and M is for Message.)

The class of the shared memory segment (R is for Resident, V is for Virtual, VX is for Virtual Extended, and M is for Message.)

blkused

The number of blocks of used memory

blkfree

The number of blocks of free memory

Virtual segment low memory reserve (bytes)

The size of reserved memory for use when critical activities are needed and the server has limited free memory, specified in bytes (You specify reserved memory in the LOW_MEMORY_RESERVE configuration parameter.)

Low memory reserve used 0 times and used maximum block size 0 bytes)

The number times that the server used the reserved memory and the maximum memory needed

onstat -g ses command: Print session-related information

Use the onstat -g ses command to display information about the session.

By default, only the DBSA can view onstat -g ses information. However, when the UNSECURE_ONSTAT configuration parameter is set to `1`, all users can view this information.

Syntax:

`onstat -gses` [*session_id*]

You can specify one of the following invocations.

onstat -g ses

Displays a one-line summary for each session

onstat -g ses *session_id*

Displays information for a specific session

Example output for all sessions

Figure 224. onstat -g ses command output

```
session                                       #RSAM    total      used       dynamic
id       user     tty      pid      hostname threads  memory     memory     explain
24       informix -        0        -        0        12288      7936       off
23       informix -        17602    carson   1        57344      48968      off
3        informix -        0        -        0        12288      9168       off
2        informix -        0        -        0        12288      7936       off
```

```
session                                       #RSAM    total      used       dynamic
id       user     tty      pid      hostname threads  memory     memory     explain
24       informix -        0        -        0        12288      7936       off
23       informix -        17602    carson   1        57344      48968      off
3        informix -        0        -        0        12288      9168       off
2        informix -        0        -        0        12288      7936       off

Last 20 Sessions Terminated

Ses ID  Username  Hostname  PID    Time              Reason
46      user_1    host_1    21220  01/19/2015.15:20  session limit txn time (60s)
43      user_1    host_1    21340  01/19/2015.15:14  session limit memory (5124 KB)
61      user_1    host_1    21404  01/19/2015.15:04  session limit logspace (10242 KB)
64      user_1    host_1    21458  01/19/2015.15:02  session limit txn time (39548 KB)
```

Output description: session section

Session id

The session ID

user

The user who started the session

tty

The tty that is associated with the front end for this session

pid

The process ID associated with the front end for this session

hostname

The hostname from which this session connected

#RSAM threads

The number of RSAM thread that is allocated for this session

total memory

The amount of memory that is allocated for this session

used memory

The amount of memory that is actually used by this session

dynamic explain

Generate explain output of the SQL statements of the session (`on` or `off`)

Output description: Last 20 Sessions Terminated section

Ses ID

The session ID

Username

The user who started the session

Hostname

The hostname from which this session connected

PID

The process ID associated with the front end for this session

Time

The time at which the session was terminated.

Reason

The limit that was exceeded, followed by the limit value in parentheses.

Example output for a specific session

Figure 225. onstat -g ses *session_id* command output for a completed SQL statement

```
session         effective                     #RSAM    total   used   dynamic
id     user     user      tty  pid    hostname threads  memory  memory explain
53     informix -         36   18638 apollo11 1        73728   63048  off

Program :
/usr/informix/bin/dbaccess

tid      name     rstcb            flags    curstk   status
77       sqlexec  4636ba20         Y--P---  4240     cond wait  sm_read   -

Memory pools    count 1
name         class addr              totalsize  freesize   #allocfrag #freefrag
53           V     4841d040         73728      10680      84         6

name           free       used           name           free       used
overhead       0          3288           scb            0          144
opentable      0          2904           filetable      0          592
log            0          16536          temprec        0          2208
gentcb         0          1656           ostcb          0          2920
sqscb          0          21296          sql            0          72
hashfiletab    0          552            osenv          0          2848
sqtcb          0          7640           fragman        0          392

sqscb info
scb              sqscb            optofc   pdqpriority optcompind  directives
481b70a0         483e2028         0        0           0           1

Sess       SQL            Current            Iso Lock       SQL  ISAM F.E.
Id         Stmt type      Database           Lvl Mode       ERR  ERR  Vers  Explain
53         -              sysmaster          CR  Not Wait   0    0    9.24  Off

Last parsed SQL statement :
  Database 'sysmaster@lx1'

Xadatasources participated in this session :
Xadatasource name                                   RMID       Active
xabasicdb@atmol10:sitaramv.xads_t3_i1                  6        YES
xabasicdb@atmol10:sitaramv.xads_t2_i1                  4        YES
xabasicdb@atmol10:sitaramv.xads_t1_i3                  3        YES
xabasicdb@atmol10:sitaramv.xads_t1_i2                  2        YES
xabasicdb@atmol10:sitaramv.xads_t1_i1                  1        YES
xabasicdb@atmol10:sitaramv.xads_t2_i2                  5        NO

DRDA client info
           Userid:
           Wrkstnname:   nemea
           Applname:     db2jcc_application
           Acctng:       JCC03510nemea
           Programid:
           Autocommit:
           Packagepath:
```

Output description: program section

Displays the full path of the client program that is used in your session. Use the client program information to monitor or stop access to the database.

Output description: threads section

Although this section has no title, the following output displays information about threads.

tid

The thread ID

name

The name of the thread

rstcb

RSAM control block

flags

Describes the status of the thread using the following codes:

Position 1

B

Waiting on a buffer

C

Waiting on a checkpoint

G

Waiting on a logical-log buffer write

L

Waiting on a lock

S

Waiting on a mutex

T

Waiting on a transaction

X

Waiting on a transaction cleanup

Y

Waiting on a condition

Position 2

*****

An asterisk in this position means that the thread encountered an I/O failure in the middle of a transaction

Position 3

A

Archive thread

B

Begin work

P

Begin Prepare or Prepared work

X

XA prepared

C

Committing or committed

R

Aborting or aborted

H

Heuristically aborted or heuristically rolling back

Position 4

P

Primary thread

Position 5

R

Reading

X

Critical section

Position 6

R

Recovery thread

Position 7

M

Monitor thread

D

Daemon thread

C

Cleaner

F

Flusher

B

B-tree scanner

curstk

Current® stack size

status

Current® thread status

Output description: memory pools header section

The information is repeated for each session pool.

name

Name of pool

class

Class of the memory where the pool is allocated from. R is for Resident, V is for Virtual, and M is for Message

addr

Address of the pool structure

totalsize

Total size of the memory that is acquired by the pool (in bytes)

freesize

Number of bytes free in the pool

#allocfrag

Number of allocated memory fragments in the pool

#freefrag

Number of free fragments in the pool

Output description: Memory pools section

name

Name of a component which allocated memory from the pool

free

Number of bytes freed

used

Number of bytes allocated

Output description: sqscb info section

scb

The session control block. This is the address of the main session structure in shared memory

sqscb

SQL level control block of the session

optofc

The current value of the **OPTOFC** environment variable or ONCONFIG configuration file setting

pdqpriority

The current value of the **PDQPRIORITY** environment variable or ONCONFIG configuration file setting

optcompind

The current value of the **OPTCOMPIND** environment variable or ONCONFIG configuration file setting

directives

The current value of the **DIRECTIVES** environment variable or ONCONFIG configuration file setting

Output description: SQL section

Displays SQL information for the specified session. This section contains the same information that is output from the onstat -g sql command. See onstat -g sql command: Print SQL-related session information on page 655.

Output description: Last parsed SQL statement section

The Last parsed SQL statement section contains the same information that is output from the onstat -g sql command. See onstat -g sql command: Print SQL-related session information on page 655.

Output description: Xadatasources participated in this session section

The Xadatasources participated in this session section shows information about the XA data sources that are available during the session, their resource manager identifiers, and whether they are currently active.

Xdatasource name

The XA data source that participated in the session

RMID

The identifier of the resource manager for the corresponding XA data source

Active

Whether the XA data source is still active

Output description: DRDA® client info section

The **DRDA® client info** section shows information about Distributed Relational Database Architecture™ (DRDA®) connections to clients.

Userid

User ID of the client user

Wrkstnname

Name of the client workstation

Applname

Name of the client application, for example `db2jcc_application`

Acctng

Accounting string from the client, for example `JCC03510nemea`

Programid

Client program identifier (not used by HCL OneDB™)

Autocommit

Default transaction autocommit mode for HCL OneDB™ data sources

Packagepath

Client package path (not used by HCL OneDB™)

Output description: Session limits section

Locks

The session's number of locks.

Memory(KB)

The session's memory.

Temp Space(KB)

The session's temporary table space.

Log Space(KB)

Log space for single transactions.

Txn Time(s)

Duration of single transactions.

Figure 226. onstat -g ses session_id command output for a running SQL statement command output

```
session           effective                       #RSAM    total    used     dynamic
id        user     user       tty  pid    hostname   threads  memory   memory   explain
37        informix -          9    13965 apollo8    1        327680   308200   off

Program :
/usr/informix/bin/dbaccess

tid       name     rstcb            flags    curstk   status
44        sqlexec  44e5b350         ---P---  4320     running-

Memory pools    count 2
name         class addr              totalsize  freesize   #allocfrag #freefrag
37           V     8c64c040          323584     18736      199        21
37*O0        V     8c756040          4096       744        1          1

name              free       used         name             free       used
overhead          0          6704         scb              0          144
opentable         0          5968         filetable        0          768
log               0          16536        temprec          0          22688
keys              0          216          ralloc           0          194672
gentcb            0          1592         ostcb            0          2992
sqscb             0          27880        sql              0          13384
hashfiletab       0          552          osenv            0          2672
sqtcb             0          9664         fragman          0          728
sapi              0          240          udr              0          272
rsam_seqscan      0          528

sqscb info
scb               sqscb            optofc   pdqpriority optcompind  directives
44ef4200          8ac90028         0        0           2           1

Sess       SQL            Current            Iso Lock       SQL  ISAM F.E.
Id         Stmt type      Database           Lvl Mode       ERR  ERR  Vers  Explain
37         SELECT         sysadmin           CR  Not Wait   0    0    9.24  Off

Current statement name : unlcur

Current SQL statement (3) :
  select * from systables, sysindexes, syscolumns

  QUERY_TIMEOUT setting: 00:00:25
  Clock time elapsed   : 00:00:13

Last parsed SQL statement :
  select * from systables, sysindexes, syscolumns
```

The QUERY_TIMEOUT on page setting and clock time are displayed only for running queries, not for DML or DDL statements or administration operations.

onstat -g shard command: Print information about the shard definition

Use the onstat -g shard command to display information about the sharding definition.

Syntax:

```
onstat -gshard
```

Output description

The output of the onstat -g shard command shows the following information without field labels.

Sharding definition name

The name of the sharding definition.

Database name

The name of the database that contains the table or collection that is distributed across multiple shards.

Table owner name

The owner of the table or collection that is distributed across multiple shards.

Table name

The name of the table or collection that is distributed across multiple shards.

Shard key

The shard key that is used for distributing rows or documents. Value can be a table column, document field, or an expression.

Sharding strategy

The method for determining which database server a new row or document is applied on. Values can be HASH (hash algorithm), CONSISTENT HASH (consistent hash algorithm), or EXPRESSION (expression).

Sharding type

Specifies source-server action after a row or document is replicated to a target server. Values can be DELETE, KEEP, or INFORMATIONAL.

Shard optimization

Specifies if queries can skip shard servers that do not contain relevant data. Values can be ENABLED or NOT ENABLED.

Version column

Specifies the column or key that is used when Enterprise Replication attempts to verify that a source row or document was not updated. The value is a column or document field.

Sharding rule

The rule for replicating data to a specific database server.

Example: Output for a sharding definition that uses consistent hash-based sharding

For this example, you have a sharding definition that was created by the following command:

```
cdr define shardCollection collection_1 database_1:john.customers_1
   --type=delete --key=b --strategy=chash --partitions=3 --versionCol=column_3
   g_shard_server_1
   g_shard_server_2
   g_shard_server_3
```

The following example shows output when the onstat -g shard command is run on **g_shard_server_1**, **g_shard_server_2**, or **g_shard_server_3**.

Figure 228. onstat -g shard command output for a sharding definition that uses a consistent hash algorithm to distribute data across multiple shard servers.

```
collection_1 database_1:john.customers_1 key:b CONSISTENT HASH:DELETE SHARD OPTIMIZATION:NOT ENABLED
Matching for delete:column_3
   g_shard_server_1 (65542) (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 4019 and 5469)
                             or (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 5719 and 6123)
                             or (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 2113 and 2652)
   g_shard_server_2 (65543) (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 6124 and 7415)
                             or (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 5470 and 5718)
                             or (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 7416 and 7873)
   g_shard_server_3 (65544) (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 2653 and 3950)
                             or mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) >= 7874
                             or mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) < 2113
                             or (mod(abs(ifx_checksum(b::LVARCHAR, 0)), 10000) between 3951 and 4018)
```

Each shard server has three hashing partitions.

Example: Output for a sharding definition that uses hash-based sharding

For this example, you have a sharding definition that was created by the following command:

```
cdr define shardCollection collection_1 database_1:josh.customers_1
   --type=delete --key=column_2 --strategy=hash --versionCol=column_3
   g_shard_server_A
   g_shard_server_B
   g_shard_server_C
   g_shard_server_D
```

The following example shows output when the onstat -g shard command is run on **g_shard_server_A**, **g_shard_server_B**, **g_shard_server_C**, or **g_shard_server_D**.

Figure 229. onstat -g shard command output for a sharding definition that uses a hash algorithm to distribute data across multiple shard servers.

```
collection_1 database_1:josh.customers_1 key:column_2 HASH:DELETE SHARD OPTIMIZATION:ENABLED
Matching for delete:column_3
g_shard_server_A (65545) mod(ifx_checksum(column_2::LVARCHAR, 0), 4) = 0
g_shard_server_B (65546) mod(ifx_checksum(column_2::LVARCHAR, 0), 4) in (1, -1)
g_shard_server_C (65547) mod(ifx_checksum(column_2::LVARCHAR, 0), 4) in (2, -2)
g_shard_server_D (65548) mod(ifx_checksum(column_2::LVARCHAR, 0), 4) in (3, -3)
```

Example: Output for a sharding definition that uses expression-based sharding

For this example, you have a sharding definition that was created by the following command:

```
cdr define shardCollection collection_2 database_2:john.customers_2
   --type=keep --key=state --strategy=expression --versionCol=version_column
   g_shard_server_F "IN ('AL','MS','GA')"
   g_shard_server_G "IN ('TX','OK','NM')"
   g_shard_server_H "IN ('NY','NJ')"
   g_shard_server_I REMAINDER
```

The following example shows output when the onstat -g shard command is run on **g_shard_server_F**, **g_shard_server_G**, **g_shard_server_H**, or **g_shard_server_I**.

Figure 230. onstat -g shard command output for a sharding definition that uses an expression to distribute data across multiple database servers.

```
collection_2 database_2:john.customers_2 key:state EXPRESSION:KEEP SHARD OPTIMIZATION:ENABLED
Matching for delete:version_column
g_shard_server_F (65564) state IN ('AL','MS','GA')
g_shard_server_G (65565) state IN ('TX','OK','NM')
g_shard_server_H (65566) state IN ('NY','NJ')
g_shard_server_I (65567) not ((state IN ('AL','MS','GA')) or (state IN('TX','OK','NM'))
or (state IN ('NY','NJ')))
```

Example: Output for a sharding definition that uses a BSON shard key and expression-based sharding

For this example, you have a sharding definition that was created by the following command:

```
cdr define shardCollection collection_3 database_3:susan.customers_3
   -t delete -k bson_value_lvarchar(data,'age') -s expression -v version
   g_shard_server_J "BETWEEN 0 and 20"
   g_shard_server_K "BETWEEN 21 and 62"
   g_shard_server_L "BETWEEN 63 and 100"
   g_shard_server_M  REMAINDER
```

The following example shows output when the onstat -g shard command is run on **shard_server_J**, **shard_server_K**, **shard_server_L**, or **shard_server_M**.

Figure 231. onstat -g shard command output for a sharding definition that uses a BSON shard key and an expression to distribute data across multiple database servers.

```
collection_3 database_3:susan.customers_3 key:bson_value_lvarchar(data,'age')
EXPRESSION:DELETE SHARD OPTIMIZATION:ENABLED
Matching for delete:version
g_shard_server_J (65568) bson_value_lvarchar(data,'age') BETWEEN 0 and 20"
g_shard_server_K (65569) bson_value_lvarchar(data,'age') BETWEEN 21 and 62"
g_shard_server_L (65570) bson_value_lvarchar(data,'age')BETWEEN 63 and 100"
g_shard_server_M (65571) not((bson_value_lvarchar(data,'age') BETWEEN 0 and 20)
or (bson_value_lvarchar(data,'age') BETWEEN 21 and 62) or (bson_value_lvarchar
(data,'age') BETWEEN 63 and 100))
```

onstat -g sle command: Print all sleeping threads

Use the onstat -g sle command to print all sleeping threads.

Syntax:

onstat -gsle

Example output

Figure 233. onstat -g sle command output

```
Current Admin VP sleep period:  10  millisecs
Sleeping threads with timeouts: 21  threads
   tid v_proc            rstcb            name     time
    49      1          b3b13a8       onmode_mon    0.02
     5      1                0  Cosvr Avail Mgr    0.05
    42      1          b3ad028      main_loop()    0.08
     9      3          b3ad6e8         xtm_svcc    0.64
    14      5                0       mgmt_thd_5    0.65
    13      4                0       mgmt_thd_4    0.65
     4      1                0       mgmt_thd_1    0.65
     6      3                0          dfm_svc    0.98
    33     13                0      mgmt_thd_13    1.54
    27     10                0      mgmt_thd_10    1.54
    21      7                0       mgmt_thd_7    1.54
    12      3                0       mgmt_thd_3    1.76
    29     11                0      mgmt_thd_11    1.76
    23      8                0       mgmt_thd_8    2.08
    31     12                0      mgmt_thd_12    2.08
    35     14                0      mgmt_thd_14    2.98
    19      6                0       mgmt_thd_6    3.00
    25      9                0       mgmt_thd_9    3.00
    37      3                0          sch_rgm    3.48
    44      5          b3af8a8      btscanner 0    7.31
    46      3          b3b0628        bum_sched   41.26
```

onstat -g smb command: Print sbspaces information

Use the onstat -g smb command to display detailed information about sbspaces.

Syntax:

onstat -gsmb [{ c | fdd | lod | s | e | h [{ cad | fdd | lod }] }]

Command	Explanation
onstat -g smb c	Lists all the chunks in the sbspace.
onstat -g smb e	Lists the entries of all smart-large-object table types.
onstat -g smb e cad	Lists the entries for the smart-large-object chunk adjunct table.
onstat -g smb e fdd	Lists the entries for the smart-large-object file descriptor table.
onstat -g smb e lod	Lists the entries in the smart-large-object header table.

Command	Explanation
onstat -g smb fdd	Lists the smart-large-object file descriptors.
onstat -g smb h	Lists the headers of all smart-large-object table types.
onstat -g smb h cad	Lists the header for the smart-large-object chunk adjunct table.
onstat -g smb h fdd	Lists the header for the smart-large-object file descriptor table.
onstat -g smb h lod	Lists the header for the smart-large-object header table.
onstat -g smb lod	Lists the header and entries in the smart-large-object header table.
onstat -g smb s	Lists the sbspace attributes (owner, name, page size, **-Df** flag settings). Fields with a value of 0 or -1 were not initialized during sbspace creation.

Example output for the onstat -g smb c command

Use the onstat -g smb c command to monitor the amount of free space in each sbspace chunk, and the size in pages of the user data and metadata. The onstat -g smb c command displays the following information for each sbspace chunk:

- Chunk number and sbspace name
- Chunk size and pathname
- Total user data pages and free user data pages
- Location and number of pages in each user-data and metadata areas

In the following example, chunk 2 of sbspace1 has 2253 original free pages (`orig fr`), 2253 user pages (`usr pgs`), and 2245 free pages (`free pg`). For the first user-data area (`Ud1`), the starting page offset is 53 and the number of pages is 1126. For the metadata area (`Md`), the starting page offset is 1179 and the number of pages is 194. For the second user data area (`Ud2`), the starting page offset is 1373 and the number of pages is 1127.

```
Chunk Summary:

sbnum 2  chunk 2
chunk:   address   flags    offset   size   orig fr   usr pgs   free pg
         303cf2a8  F------- 0        2500   2253      2253      2245
         path: /usr11/myname/sbspace1

         start pg   npages
Ud1  :   53         1126
Md   :   1179       194
Ud2  :   1373       1127
```

Output for the onstat -g smb s command

The onstat -g smb s command displays the storage attributes for all sbspaces in the system:

- sbspace name, flags, owner
- logging status
- average smart-large-object size

- first extent size, next extent size, and minimum extent size
- maximum I/O access time
- lock mode

For more information on the onstat -g smb command, see the *HCL OneDB™ Performance Guide*.

onstat -g smx command: Print multiplexer group information

Use the onstat -g smx command to display information about the server multiplexer group for servers using SMX.

Syntax:

```
onstat -gsmx [ ses ]
```

Command	Explanation
onstat -g smx	Displays SMX connection statistics
onstat -g smx ses	Displays SMX session statistics

Example output

Figure 236. onstat -g smx command output

```
SMX connection statistics:
SMX control block: 0x47d5e028

  Peer server name: lx1
  SMX connection address: 0x47d60d10
  Encryption status: Disabled
  Total bytes sent: 27055
  Total bytes received: 2006989
  Total buffers sent: 782
  Total buffers received: 7090
  Total write calls: 782
  Total read calls: 7090
  Total retries for write call: 0
  Data compression level: 1
  Data sent: compressed 40760 bytes by 33%
  Data received: compressed 12579324 bytes by 84%
```

Output description

SMX control block

SMX control block

Peer server name

Displays the name of the peer server

SMX connection address

Displays the address of the SMX connection

Encryption status

Displays whether encryption is enabled or disabled

Total bytes sent

Displays the total number of bytes sent

Total bytes received

Displays the total number of bytes received

Total buffers sent

Displays the total number of buffers sent

Total buffers received

Displays the total number of buffers received

Total write calls

Displays the total number of write calls

Total read calls

Displays the total number of read calls

Total retries for write call

Displays the total number of retries for write call

Data compression level

Displays the SMX compression level as set by the SMX_COMPRESS configuration parameter

Data sent: compressed x bytes by y%

Displays the uncompressed number of bytes and the compression ratio of the data sent

Data received: compressed x bytes by y%

Displays the uncompressed number of bytes and the compression ratio of the data received

Example Output

Figure 237. onstat -g smx ses Output

```
SMX session statistics:
SMX control block: 0x17c69028

Peer        SMX session        client        reads        writes
name        address            type
delhi_sec     19022050   smx Clone Send           6          183
```

Output Description

SMX control block

SMX control block

Peer name

Displays the name of the peer server

SMX session address

SMX session address

Client type

Displays type of secondary server

reads

Displays the total number of session reads

writes

Displays the total number of session writes

onstat -g spi command: Print spin locks with long spins

Use the onstat -g spi command to display information about spin locks with long spins.

Syntax:

```
onstat  -gspi
```

Many resources in the server are accessed by two or more threads. In some of these accesses (such as updating a shared value), the server must guarantee that only one thread is accessing the resource at a time. A *spin lock* is the mechanism used to provide this mutually exclusive access for some resources. With this type of lock, a thread that did not succeed in acquiring the lock on the first try (because another thread was holding it) repeatedly attempts to acquire the lock until it succeeds.

The overhead cost of a spin lock is small, and spin locks are normally used for resources that require mutual exclusion for short periods of time. However, if a spin lock becomes highly contended, the loop-and-retry mechanism can become expensive.

The onstat -g spi command is helpful for identifying performance bottlenecks that are caused by highly contended spin locks. This option lists spin locks with waits, those spin locks for which a thread was not successful in acquiring the lock on its first attempt and thus had to loop and re-attempt.

Example output

Figure 239. onstat -g spi command output

```
Spin locks with waits:

Num Waits   Num Loops   Avg Loop/Wait    Name
114         117675      1032.24          lockfr3
 87         256461      2947.83          fast mutex, lockhash[832]
  1             11        11.00          fast mutex, 1:bhash[16668]
  4          51831     12957.75          fast mutex, 1:lru-4
  1            490       490.00          fast mutex, 1:bf[994850] 0xe00002 0x14eb32000
```

Output description

Num Waits (decimal)

Total number of times a thread waited for this spin lock.

Num Loops (decimal)

Total number of attempts before a thread successfully acquired the spin lock.

Avg Loop/Wait (floating point)

Average number of attempts needed to acquire the spin lock. Computed as `Num Loops / Num Waits`.

Name (string)

Uses the following codes to name the spin lock

lockfr

The lock free list. The number after **lockfr** is the index into the lock free list array.

lockhash[]

The lock hash bucket. The field inside the brackets is the index into the lock hash bucket array.

:bhash []

The buffer hash bucket. The field before the colon is the buffer pool index; the field inside the brackets after **bhash** is the index into the buffer hash bucket array.

:lru-

The LRU latch. The field before the colon is the buffer pool index; the field after **lru-** identifies the buffer chain pairs that are being used.

:bf[]

The buffer latch. The field before the colon is the buffer pool index; the field inside the brackets after **bf** is the position of buffer in the buffer array. The next two fields are the partition number and the page header address in memory for the buffer in hex form.

onstat -g sql command: Print SQL-related session information

Use the onstat -g sql command to display SQL-related information about a session.

By default, only the DBSA can view onstat -g sql syssqltrace information. However, when the UNSECURE_ONSTAT configuration parameter is set to `1`, all users can view this information.

Syntax:

`onstat -gsql` *sessionid*

You can specify one of the following invocations.

Invocation

Explanation

onstat -g sql

Displays a one line summary for each session

onstat -g sql *sessionid*

Displays SQL information for a specific session

Note: Encrypted passwords and password hint parameters in encryption functions are not shown. The following figure displays an encrypted password in the `Last parsed SQL statement` field.

Figure 241. onstat -g sql command output for a completed SQL statement

```
onstat -g sql 22

Sess  SQL            Current            Iso Lock       SQL  ISAM F.E.          Current
Id    Stmt type      Database           Lvl Mode       ERR  ERR  Vers Explain    Role
22     -             test               CR  Not Wait    0    0   9.03 Off         hr
Last parsed SQL statement :
  select id, name, decrypt_char(ssn, 'XXXXXXXXXX') from emp
```

Output description

Sess id

The session identifier

SQL Stmt type

The type of SQL statement

Current® Database

Name of the current database of the session

ISO Lvl

Isolation level

DR

Dirty Read

CR

Committed Read

CS

Cursor Stability

DRU

Dirty Read, Retain Update Locks

CRU

Committed Read, Retain Update Locks

CSU

Cursor Stability, Retain Update Locks

LC

Committed Read, Last Committed

LCU

Committed Read Last Committed with Retain Update Locks

RR

Repeatable Read

NL

Database Without Transactions

Lock mode

Lock mode of the current session

SQL Error

SQL error number encountered by the current statement

ISAM Error

ISAM error number encountered by the current statement

F.E. Version

The version of the SQLI protocol used by the client program

Explain

SET EXPLAIN setting

Current® Role

Role of the current user

Figure 242. onstat -g sql command output for a running SQL statement statement

```
onstat -g sql 28

Sess  SQL        Current     Iso Lock      SQL  ISAM F.E.
Id    Stmt type  Database    Lvl Mode      ERR  ERR  Vers  Explain
28    SELECT     sysmaster   CR  Not Wait 0    0    9.24  Off

Current statement name : unlcur

Current SQL statement (8) :
  select * from systables, syscolumns, sysindexes

  QUERY_TIMEOUT setting:     0 (No Timeout)
  Clock time elapsed   : 00:00:12

Last parsed SQL statement :
  select * from systables, syscolumns, sysindexes
```

The QUERY_TIMEOUT setting and clock time are displayed only for running queries, not for DML or DDL statements or administration operations.

onstat -g spf: Print prepared statement profiles

Use the **onstat -g spf** command to display current statistics about SQL queries.

You can use the statistics to determine the cost of each statement.

Syntax:

`onstat -gspf`

If SQL tracing is enabled, the information that is shown is a snapshot of the work that is completed by the statement and might change as the statement continues to run. For example, to monitor the growth rate of buffer reads or writes in an active statement, you can issue three **onstat -g spf** runs at 2-second intervals.

If SQL tracing is disabled, a warning message is issued: `"Statistics disabled"`.

Example output

Figure 244. **onstat -g spf** command output

```
Statement profiles
sid  sdb      tottm  execs runtm  pdq scans sorts bfrd  pgrd  bfwrt  pgwrt  lkrqs  lkwts
35   4de84028 0.01   0     0.01   0   0     0     301   352   0      512    2998   0
25   4dc0b028 0.00   0     0.00   0   0     0     0     0     0      0      0      0
...
```

Output description

sid

The session ID.

sdb

The last 8 digits of the statement pointer.

tottm

The current total run time, in seconds, of all statements.

execs

The current number of completed statement runs. This value does not include statements that are running.

runtm

The current run time of the statement, in seconds.

pdq

The current parallel database queries (PDQ) priority level. The PDQ priority value can be any integer from 0 through 100.

scans

The current number of PDQ scans that are allocated.

sorts

The current number of completed sorts.

bfrd

The current number of buffer reads.

pgrd

The current number of page reads.

bfwrt

The current number of buffer writes.

pgwrt

The current number of page writes.

lkrqs

The current number of lock requests.

lkwts

The current number of lock waits.

onstat -g src command: Patterns in shared memory

Use the onstat -g src command to search for patterns in shared memory.

Syntax:

`onstat -gsrc` *patternmask*

Example output

The following example shows output for the onstat -g src***pattern mask*** command where *pattern* = `0x123` and *mask* = `0xffff`.

Figure 246. onstat -g src command output

```
Search Summary:
addr               contents
000000000ad17a50: 01090000 00000000 00000000 00000123   ........ .......#
000000000ad7dec0: 00000001 014e3a0c 00000000 0ade0123   .....N:. .......#
```

Output description

addr (hexadecimal)

Address in shared memory where search pattern is found

contents (hexadecimal)

Contents of memory at given address

onstat -g ssc command: Print SQL statement occurrences

Use the onstat -g ssc command to monitor the number of times that the database server reads the SQL statement in the cache.

By default, only the DBSA can view onstat -g ssc syssqltrace information. However, when the UNSECURE_ONSTAT configuration parameter is set to `1`, all users can view this information.

Syntax:

```
onstat -gssc [{all | pool}]
```

The **all** option reports the *key-only* cache entries as well as the fully cached statements. If the value in the **hits** column is less than the STMT_CACHE_HITS value, that entry is a *key-only* cache entry. For more information, see memory utilization in the *HCL OneDB™ Performance Guide*.

The **pool** option reports usage of all memory pools for the SQL statement cache. The output displays information on the name, class, address, and total size of the memory pools. For more information, see improving query performance in the *HCL OneDB™ Performance Guide*.

Example output

Figure 248. onstat -g ssc command output

```
Statement Cache Summary:
#lrus   currsize  maxsize   Poolsize  #hits   nolimit
4       117640    524288    139264    0       1
Statement Cache Entries:
lru hash ref_cnt hits flag heap_ptr     database           user
----------------------------------------------------------------
  0  262       0    7   -F  aad8038      sscsi007           admin
   INSERT INTO ssc1 ( t1_char , t1_short , t1_key , t1_float , t1_smallfloat
    , t1_decimal , t1_serial ) VALUES ( ? , ? , ? , ? , ? , ? , ? )
  0  127       0    9   -F  b321438      sscsi007           admin
   INSERT INTO ssc2 ( t2_char , t2_key , t2_short ) VALUES ( ? , ? , ? )
  1  134       0   15   -F  aae0c38      sscsi007           admin
   SELECT t1_char , t1_short , t1_key , t1_float , t1_smallfloat ,
    t1_decimal , t1_serial FROM ssc1 WHERE t1_key = ?
  1  143       0    3   -F  b322c38      sscsi007           admin
   INSERT INTO ssc1 ( t1_char , t1_key , t1_short ) SELECT t2_char , t2_key
    + ? , t2_short FROM ssc2
  2   93       0    7   -F  aae9838      sscsi007           admin
   DELETE FROM ssc1 WHERE t1_key = ?
  2  276       0    7   -F  aaefc38      sscsi007           admin
   SELECT count ( * ) FROM ssc1
  2  240       1    7   -F  b332838      sscsi007           admin
   SELECT COUNT ( * ) FROM ssc1 WHERE t1_char = ? AND t1_key = ? AND
    t1_short = ?
  3   31       0    7   -F  aaec038      sscsi007           admin
   SELECT count ( * ) FROM ssc1 WHERE t1_key = ?
  3   45       0    1   -F  b31e438      sscsi007           admin
   DELETE FROM ssc1
  3  116       0    0   -F  b362038      sscsi007           admin
   SELECT COUNT ( * ) FROM ssc1
    Total number of entries: 10.
```

Output description - Statement Cache Summary section

#lrus

Number of least recently used queues (LRUS)

currsize

Current® cache size

maxsize

Limit on total cache memory

Poolsize

Total pool size

#hits

The number of hits before insertion. This number equals the value of the STMT_CACHE_HITS configuration parameter

nolimit

The value of the STMT_CACHE_NOLIMIT configuration parameter

Output description - Statement Cache Entries section

The Statement Cache Entries section shows the entries that are fully inserted into the cache.

lru

The index of lru queue to which the cache entry belongs

hash

Hash values of cached entry

ref_count

Number of threads referencing the statement

hits

Number of times a statement matches a statement in the cache. The match can be for a key-only or fully cached entry.

flag

Cache entry flag `-D` indicates that the statement is dropped, `-F` indicates that the statement is fully cached, and `-I` indicates that the statement is in the process of being moved to a fully cached state

heap_ptr

Address of memory heap for cache entry

onstat -g stk command: Print thread stack

Use the onstat -g stk *tid* command to display the stack of the thread specified by thread ID.

This option is not supported on all platforms and is not always accurate.

Syntax:

onstat -g stk *tid*

Example output

Figure 250. onstat -g stk *tid* command output

```
Stack for thread: 2 adminthd
 base: 0x000000010aad5028
  len:   33280
   pc: 0x00000001002821e8
  tos: 0x000000010aadc621
state: running
   vp: 2

0x1002821e8 oninit :: yield_processor + 0x260 sp=0x10aadce20(0x10ac834d0, 0x0, 0x1,
   0x100000000, 0xc8a000, 0x100c8a000)
0x100274e38 oninit :: wake_periodic + 0xdc sp=0x10aadced0 delta_sp=176(0x41b0, 0xc7a024bc,
   0x0, 0x41c4, 0x10aacf598, 0x90)
0x100274fcc oninit :: admin_thread + 0x108 sp=0x10aadcf80 delta_sp=176(0x0, 0x2328,
   0xd26c00, 0x5, 0xc8a000, 0x156c)
0x1002484ec oninit :: startup + 0xd8 sp=0x10aadd050 delta_sp=208(0xa, 0x10aad47d0,
   0x10aad47d0, 0x100db1988, 0xd1dc00, 0x1)
```

onstat -g stm command: Print SQL statement memory usage

Use the onstat -g stm command to display the memory that each prepared SQL statement uses.

By default, only the DBSA can view onstat -g stm syssqltrace information. However, when the UNSECURE_ONSTAT configuration parameter is set to `1`, all users can view this information.

Syntax:

onstat -gstm

To display the memory for only one session, specify the session ID in the onstat -g stm command.

Example output

Figure 252. onstat -g stm command output

```
session   65 ----------------------------------------------------------
 sdblock  heapsz  statement ('*' = Open cursor)
 aad8028   16544   SELECT COUNT ( * ) FROM ssc1 WHERE t1_char = ?
 AND t1_key = ? AND t1_short = ?
```

Output description

sdblock

Address of the statement descriptor block

heapsz

Size of the statement memory heap

statement

Query text

onstat -g stq command: Print queue information

Use the onstat -g stq command to display information about the queue.

Syntax:

`onstat -gstq`*session*

To view queue information for a particular session specify the *session* option. Omit the *session* option to view queue information for all sessions.

Example output

Figure 254. onstat -g stq command output

```
Stream Queue: (session 25 cnt 4)  0:db12400  1:db18400  2:dcf0400  3:dcf6400
 Full Queue:  (cnt 2  waiters 0)  0:0  1:db12400
 Empty Queue: (cnt 0  waiters 0)
```

Output description

session

Session id

cnt

Number of stream queue buffers

waiters

Number of threads waiting for the stream queue buffer

onstat -g sts command: Print stack usage for each thread

Use the onstat -g sts command to display information about the maximum and current stack use for each thread.

Syntax:

`onstat -gsts`

Example output

Figure 256. onstat -g sts command output

```
Stack usage:

TID     Total   Max          Current        Thread Name
                bytes   %    bytes   %
2       32768   3124    9    3079    9      adminthd
3       32768   2870    8    2871    8      childthd
5       32768   14871   45   2871    8      Cosvr Avail Mgr
6       32768   2870    8    2871    8      dfm_svc
7       131072  3190    2    3191    2      xmf_svc
9       32768   3126    9    3127    9      xtm_svcc
10      32768   3580    10   3335    10     xtm_svcp
11      32768   3238    9    3239    9      cfgmgr_svc
12      32768   6484    19   2871    8      lio vp 0
14      32768   6484    19   2871    8      pio vp 0
16      32768   6484    19   2871    8      aio vp 0
18      131072  10391   7    2871    2      msc vp 0
20      32768   4964    15   2871    8      fifo vp 0
22      32768   4964    15   2871    8      fifo vp 1
24      32768   6028    18   2871    8      aio vp 1
26      32768   5444    16   2951    9      dfmxpl_svc
27      32768   2886    8    2887    8      sch_svc
28      32768   7812    23   5015    15     rqm_svc
29      32768   7140    21   3079    9      sm_poll
30      32768   11828   36   6439    19     sm_listen
31      32768   2870    8    2871    8      sm_discon
32      32768   14487   44   4055    12     main_loop()
33      32768   4272    13   2903    8      flush_sub(0)
34      32768   2902    8    2903    8      flush_sub(1)
35      32768   2870    8    2871    8      btscanner 0
36      32768   3238    9    3239    9      aslogflush
37      32768   3055    9    2887    8      bum_local
38      32768   3238    9    3239    9      bum_rcv
39      32768   4902    14   4903    14     onmode_mon
42      32768   4964    15   2871    8      lio vp 1
44      32768   5136    15   2871    8      pio vp 1
```

onstat -g sym command: Print symbol table information for the oninit utility

Use the onstat -g sym command to display symbol table information for the oninit utility.

Syntax:

onstat -gsym

Example output

Figure 258. onstat -g sym command output

The following example shows the first few lines from the output:

```
Table for oninit has 23378 entries
    Initial value for -base-: 0x0
        0x3451e0 _start
        0x345300 .ld_int
        0x345348 .ld_llong
        0x3453dc .ld_float
        0x345428 .ld_double
        0x3454c4 .st_int
        0x3454fc .st_llong
        0x34556c .st_float
        0x3455c0 .st_double
        0x34565c .st_float_foreff
        0x345694 .st_double_foreff
        0x345718 main
        0x34c2ac get_cfgfile
        0x34c2fc is_server_alias
```

Output description

The onstat -g sym command displays the relative in-memory address and name of symbols (functions and variables) in the oninit utility.

onstat -g tpf command: Print thread profiles

Use the onstat -g tpf command to display thread profiles.

Syntax:

onstat -gtpf*tid*

Specify the *tid* thread ID to print the profile for a specific thread. Set *tid* to 0 to display the profiles for all of the threads.

Example output

Figure 260. onstat -g tpf command output

```
onstat -g tpf 945

Thread profiles
tid lkreqs lkw dl to lgrs isrd iswr isrw isdl isct isrb lx bfr bfw lsus lsmx seq
945 1969   0   0  0  6181 1782 2069 13   0    0    0    0  16183 7348 743580 0    6
```

Output description

tid

Thread ID

lkreqs

Lock requests

lkw

Lock waits

dl

Deadlocks

to

Remote deadlock timeout

lgrs

Log records

isrd

Number of reads

iswr

Number of writes

isrw

Number of rewrites

isdl

Number of deletes

isct

Number of commits

isrb

Number of rollbacks

lx

Long transactions

bfr

Buffer reads

bfw

Buffer writes

lsus

Log space currently used

lsmx

Max log space used

seq

Sequence scans

onstat -g ufr command: Print memory pool fragments

Use the onstat -g ufr command to display a list of the fragments that are currently in use in the specified memory pool.

This command requires an additional argument to specify either a pool name or session ID whose memory pool information is to be displayed. Each session is allocated a memory pool with the same name as the session ID. Use the onstat -g mem command to identify the pool name and the onstat -g ses command to identify the session ID.

Syntax:

onstat -gufr { *pool name* | **sessionid** }

Memory pools are broken into fragments for various uses. With the onstat -g ufr command it is possible to see a list of these fragments showing their respective sizes in bytes and the type of information they contain. The information provided is generally used by Technical Support to assist in the analysis of a reported problem.

Example output for a specified pool name

Figure 262. onstat -g ufr global command output for a specified pool name

```
Memory usage for pool name global:
size       memid
1736       overhead
23544      mcbmsg
72         messages
33112      osenv
25432      rsam
88         shmblklist
5170664    net
```

Example output for a specified session ID

The following example shows the output for session ID 6.

Figure 263. onstat -g ufr command output for a specified session ID

```
Memory usage for pool name 6:
size       memid
3256       overhead
144        scb
2968       ostcb
18896      sqscb
3312       opentable
72         sql
808        filetable
352        fragman
552        hashfiletab
1584       gentcb
12096      log
2960       sqtcb
2928       osenv
720        keys
224        rdahead
16248      temprec
```

Output description

size (decimal)

Size, in bytes, of the pool fragment.

memid (string)

Name of the pool fragment.

onstat -g vpcache command: Print CPU virtual processor and tenant virtual processor private memory cache statistics

Run the onstat -g vpcache command to display statistics about CPU virtual processor and tenant virtual processor private memory caches.

Syntax:

```
onstat  -gvpcache
```

Example output

The output for each CPU or tenant virtual processor has the same format. The following example shows the output for one CPU virtual processor.

Figure 265. onstat -g vpcache command output

```
CPU virtual processor memory block cache statistics - 4096 byte blocks

Number of 4096 byte memory blocks requested for each CPU virtual processor:262144
CPU virtual processor memory block cache mode : Dynamic

vpid    pid         Blocks held  Hit percentage   Free cache
1       2557540     4667202       99.2 %           100.0 %

Current total virtual processor allocations from cache: 59466799, Total frees: 60209953

  size  cur blks   tgt blks   alloc      miss     free      drain  draintime
  1     1662023    9661       49167485   0        49816526  0      Thu Apr 11 09:43:35 2013
  2     130        52428      7609556    297043   7609612   0      Thu Jan  1 00:00:00 1970
  3     329160     9          905094     0        943256    0      Thu Apr 11 09:43:36 2013
  4     424        9          306637     16192    306506    0      Thu Apr 11 09:43:33 2013
  5     10         9          119313     122607   119315    0      Thu Apr 11 09:43:36 2013
  6     20790      9          55305      0        57700     0      Thu Apr 11 09:43:23 2013
  7     9877       9          31164      0        31942     0      Thu Apr 11 09:43:14 2013
  8     2816       5242       6500       0        6537      0      Thu Jan  1 00:00:00 1970
  9     234        9          606575     8323     605525    0      Thu Apr 11 09:43:36 2013
  10    1130       9          5597       0        5679      0      Thu Apr 11 09:43:18 2013
  11    231        5242       1808       0        1753      0      Thu Jan  1 00:00:00 1970
  12    1068       9          5667       0        5666      0      Thu Apr 11 09:43:28 2013
  13    65         5242       7114       175      7110      0      Thu Jan  1 00:00:00 1970
  14    28         5242       26200      172      26185     0      Thu Jan  1 00:00:00 1970
  15    30         5242       13562      553      13547     0      Thu Jan  1 00:00:00 1970
  16    2627136    34         349124     0        408425    0      Thu Apr 11 09:43:35 2013
  17    1309       9          59         0        107       0      Thu Apr 11 09:27:33 2013
  18    198        5242       7          0        6         0      Thu Jan  1 00:00:00 1970
  19    190        5242       5          0        1         0      Thu Jan  1 00:00:00 1970
  20    60         5242       30         19       19        0      Thu Jan  1 00:00:00 1970
  21    462        5242       38         0        43        0      Thu Jan  1 00:00:00 1970
  22    22         5242       3          0        1         0      Thu Jan  1 00:00:00 1970
  23    69         5242       141        15       135       0      Thu Jan  1 00:00:00 1970
  24    4944       35         189509     2078     185347    0      Thu Apr 11 09:43:35 2013
  25    75         5242       1          0        1         0      Thu Jan  1 00:00:00 1970
  26    0          9          364        220      361       0      Thu Apr 11 09:39:17 2013
  27    27         5242       1          0        2         0      Thu Jan  1 00:00:00 1970
  28    56         5242       415        33       410       0      Thu Jan  1 00:00:00 1970
  29    319        5242       7101       735      7088      0      Thu Jan  1 00:00:00 1970
  30    3240       5242       174        0        223       0      Thu Jan  1 00:00:00 1970
  31    279        11         51994      2515     50682     0      Thu Apr 11 09:43:36 2013
  32    800        5242       256        0        243       0      Thu Jan  1 00:00:00 1970
```

Output description

vpid

The ID of the virtual processor

pid

The process ID for the virtual processor that is assigned by the operating system

Blocks held

The number of 4096 byte blocks that are available in the private memory cache

Hit percentage

The percentage of time that a block was available when requested

Free cache

The percentage of time that blocks were freed for reuse without being drained

Current VP total allocations from cache

The number of times a block or group of blocks was taken from the cache

Total frees

The number of times a block or group of blocks was added to the cache

size

The size of the memory blocks, in 4096-byte blocks

cur blks

The current number of 4096-byte blocks that are allocated (a multiple of `size`)

tgt blks

The target number of blocks for the cache entry before the cache is drained

alloc

The number of times a requestor received a block of this size

miss

The number of times a block was requested but none were available

free

The number of times a memory block was placed into the cache

drain

The number of times an aged block was forced out to make room for another block

draintime

The last time the bin of memory blocks was drained

onstat -g wai command: Print wait queue thread list

Use the onstat -g wai command to display a list of the threads in the system that are currently in the wait queue and not currently executing. The output is sorted by thread ID.

Syntax:

```
onstat  -gwai
```

Example output

Figure 267. onstat -g wai command output

```
Waiting threads:
 tid     tcb             rstcb           prty status                vp-      name
 2       46b1ea40         0               1    IO Idle                 5lio       lio vp 0
 3       46b3dc58         0               1    IO Idle                 6pio       pio vp 0
 4       46b5dc58         0               1    IO Idle                 7aio       aio vp 0
 5       46b7cc58         0               1    IO Idle                 8msc       msc vp 0
 6       46b1ed10         460f5028        1    sleeping secs: 1        3cpu       main_loop()
 9       46d0d6e0         0               1    sleeping forever        1cpu       soctcplst
 10      46d70b48         0               1    sleeping forever        3cpu       sm_listen
 11      46e5d9a0         0               1    sleeping secs: 1        3cpu       sm_discon
 12      46e5dc70         460f5820        1    sleeping secs: 1        3cpu       flush_sub(0)
 13      46e8a5a8         460f6018        1    sleeping secs: 1        3cpu       aslogflush
 14      46fe8148         460f6810        1    sleeping secs: 41       3cpu       btscanner_0
 15      46fe84a8         0               1    IO Idle                10aio       aio vp 1
 16      46fe8778         460f7008        1    sleeping secs: 1        1cpu       onmode_mon
 36      47531960         460f7ff8        1    sleeping secs: 253      3cpu       dbScheduler
 37      47531c30         460f87f0        1    sleeping forever        4cpu       dbWorker1
 38      47491028         460f7800        1    sleeping forever        4cpu       dbWorker2
```

Output description

tid (decimal)

Thread ID

tcb (hex)

In-memory address of the thread control block

rstcb (hex)

In-memory address of the RSAM thread control block

prty (decimal)

Thread priority. Higher numbers represent higher priorities

status (string)

Current® status of the thread

vp- (decimal and string)

Virtual processor integer ID of the VP on which the thread last ran, concatenated with the name of the VP upon which the thread runs

name (string)

Name of the thread

onstat -g wmx command: Print all mutexes with waiters

Use the onstat -g wmx command to display all of the mutexes with waiters.

Syntax:

```
onstat  -gwmx
```

Example output

Figure 269. onstat -g wmx command output

```
Mutexes with waiters:
mid       addr              name                 holder    lkcnt  waiter    waittime
134825    7000002043a9148   free_lock            11009     0      200       22921
                                                                  11010     22918
```

Output description

mid

Internal mutex identifier

addr

Address of locked mutex

name

Name of the mutex

holder

Thread ID of the thread that is holding the mutex

0 = The read/write mutex is held in shared mode

lkcnt

For a read/write mutex, the current number of threads that are locking the mutex in shared mode. For a relockable mutex, the number of times the mutex was locked or relocked by the thread that is holding the mutex.

The number of times the mutex was locked or relocked by the thread that is holding the mutex

waiter

List of IDs of the threads that are waiting for this mutex

waittime

Amount of time in seconds that the thread is waiting

onstat -g wst command: Print wait statistics for threads

Use the onstat -g wst command to show the wait statistics for the threads within the system.

The WSTATS configuration parameter must be set to `1` to enable wait statistics collection. For more information, see WSTATS configuration parameter on page 216 .

Syntax:

```
onstat  -gwst
```

Example output

```
Version 11.70.F  -- On-Line -- Up 18:52:59 -- 78856 Kbytes
name   tid    state           n          avg(us)    max(us)
msc vp 0 5    ready           6               9         17
msc vp 0 5    run             6            1107       2215
msc vp 0 5    IO Idle         5           2985.9s   1496.1s

main_loo 7    IO Wait         55           6496      16725
main_loo 7    yield time      44929         1.2s     343.1s
main_loo 7    ready           44998      206085      343.1s
main_loo 7    run             44985           5        436

...

sqlexec  63   IO Wait         2            1118       2165
sqlexec  63   other cond      6           34237     204142
sqlexec  63   ready           9               7         16
sqlexec  63   run             7             1.1s       7.7s
```

Output description

name (string)

Thread name

tid (decimal)

Thread ID

state (string)

State the thread waited in for this line of output. A single thread can have multiple lines of output if it waited in more than one state. Values that can appear in the `state` field include:

`chkpt cond`: The thread waited for a checkpoint condition.
`cp mutex`: The thread waited for checkpoint mutex to become available.
`deadlock mutex`: The thread waited for a deadlock mutex to become available.
`empty Q`: The thread waited for an empty buffer on a queue.
`fork`: The thread waited for a child thread to run.
`full Q`: The thread waited for a full buffer on a queue.
`IO Idle`: The I/O thread was idle.
`IO Wait`: The thread yielded while it waited for I/O completion.
`join wait`: The thread waited for another thread to exit.
`lock mutex`: The thread waited for lock mutex to become available.
`lockfree mutex`: The thread waited for a lock-free mutex to become available.
`logflush`: Logical log flushing occurred.

`log mutex`: The thread waited for logical log mutex to become available.

`logcopy cond`: The thread waited for logical log copy condition.

`logio cond`: The thread waited for a logical log condition.

`lrus mutex`: The thread waited for a buffer LRU mutex to become available.

`misc`: The thread waited for a miscellaneous reason.

`other cond`: The thread waited for an internal condition.

`other mutex`: The thread waited for an internal system mutex to become available.

`other yield`: The thread yielded for an internal reason.

`OS read`: The thread waited for an operating system read call to complete.

`OS write`: The thread waited for an operating system write call to complete.

`ready`: The thread was ready to run.

`run`: The thread ran.

`sort io`: The thread waited for sort I/O completion.

`vp mem sync`: The thread waited for synchronization of virtual processor memory.

`yield bufwait`: The thread yielded while it waited for a buffer to become available.

`yield 0`: The thread yielded with an immediate timeout.

`yield time`: The thread yielded with a timeout.

`yield forever`: The thread yielded and stays that way until it wakes up.

n (decimal)

Number of times the thread waited in this state

avg(us) (floating point)

Average user time the thread spent waiting in this state per wait occurrence. Time is in microseconds; an `s` after the value indicates user time in seconds.

max(us) (floating point)

Maximum user time the thread spent waiting in this state for a single wait occurrence. Time is in microseconds; an `s` after the value indicates user time in seconds.

onstat -G command: Print TP/XA transaction information

Use the onstat -G command to display information about global transactions generated through the TP/XA library.

Syntax:

```
onstat -G
```

Example output

Figure 272. onstat -G command output

```
Global Transaction Identifiers
address    flags  isol    timeout  fID       gtl  bql  data      dbpartnum
45cb0318   -LH-G  COMMIT  0        4478019   2    2    30323032  100163
```

For a tightly coupled transaction, all branches share the same transaction address shown in the address column.

Output description

address

Transaction address

flags

Flag codes for position 1 (current transaction state):

A

User thread attached to the transaction

S

TP/XA suspended transaction

C

TP/XA waiting for rollback

Flag codes for position 2 (transaction mode):

T

Tightly-coupled mode (MTS)

L

Loosely-coupled mode (default mode)

Flag codes for position 3 (transaction stage):

B

Begin work

P

Distributed query prepared for commit

X

TP/XA prepared for commit

C

Committing or committed

R

Rolling back or rolled back

H

Heuristically rolling back or rolled back

Flag code for position 4:

X

XA data source global transaction

Flag codes for position 5 (type of transaction):

G

Global transaction

C

Distributed query coordinator

S

Distributed query subordinate

B

Both distributed query coordinator and subordinate

M

Redirected global transaction

isol

Transaction isolation level

timeout

Transaction lock timeout

fID

Format ID

gtl

Global transaction ID length

bql

Branch qualifier length

data

Transaction-specific data

dbpartnum

Database identifier of where the transaction starts

onstat -h command: Print buffer header hash chain information

Use the onstat -h command to display information about the buffer header hash chains (sometimes called "hash buckets") that are used to access pages in each buffer pool.

Syntax:

```
onstat  -h
```

Example output

The output is displayed in the form of a numeric histogram of chain lengths, with summary information for each buffer pool. All numeric values in the output are decimal. Shorter hash chains enable requested buffers to be located more quickly by the server, because on average it will need to check fewer buffer headers on a target chain to find the target buffer.

The page size of the buffer pool in bytes is shown as a header to the output for each buffer pool. The histogram and summary information are then presented for that buffer pool.

Figure 274. onstat -h command output

```
Buffer pool page size: 2048

buffer hash chain length histogram
   # of chains         of len
          3423              0
          4546              1
           223              2
          8192   total chains
          4992   hashed buffs
          5000    total buffs

Buffer pool page size: 4096

buffer hash chain length histogram
   # of chains         of len
           707              0
           315              1
             2              2
          1024   total chains
           319   hashed buffs
          1000    total buffs
```

Output description

Histogram Information on Hash Chains

The histogram information has a row for each buffer hash chain length that presently exists in the system. Each row has two columns:

# of chains

Number of hash chains of the given length

of len

Length of these chains

Summary Information Per Buffer Pool

total chains

Number of hash chains that exist for this buffer pool

hashed buffs

Number of buffer headers currently hashed into the hash chains for this buffer pool

total buffs

Total number of buffers in this buffer pool

onstat -i command: Initiate interactive mode

Use the onstat -i command to put the onstat utility in the interactive mode.

Syntax:

`onstat -i` [{ `r`*seconds* | `rz`*seconds* }]

In interactive mode, you can enter multiple onstat options per session, but only one at a time. An onstat prompt appears and allows you to enter an option.

Important: In interactive mode, do not precede the option with a dash.

Additional options

Two additional options, onstat r ***seconds*** and onstat rz ***seconds***, are available in interactive mode. The onstat r ***seconds*** option is similar to the current onstat -r ***seconds*** option, which repeatedly generates a display. If an administrator executes onstat r ***seconds*** at the interactive-mode prompt, the prompt changes to reflect the specified interval in seconds and reappears, waiting for the next command. In the following example, the display generated by the next command repeats every three seconds:

```
onstat> r 3
onstat[3]>
```

The onstat rz ***seconds*** option enables you to repeat the next command as specified and set all profile counters to 0 between each execution.

Terminating interactive mode or repeating sequence

To terminate the interactive mode, press `CTRL-d`.

To terminate a repeating sequence, press `CTRL-c`.

onstat -k command: Print active lock information

Use the onstat -k command to print information about active locks, including the address of the lock in the lock table.

Syntax:

```
onstat -k
```

The maximum number of locks available is specified by the value of the LOCKS configuration parameter in the `onconfig` file.

Example output

Figure 277. onstat -k command output

```
Locks
address  wtlist   owner    lklist   type     tblsnum  rowid    key#/bsiz
a095f78  0        a4d9e68  0        HDR+S    100002   203         0
 1 active, 2000 total, 2048 hash buckets, 0 lock table overflows
```

In the following output, the number 2 in the last row shows an Enterprise Replication pseudo lock:

```
Locks
address  wtlist    owner    lklist   type    tblsnum  rowid    key#/bsiz
a1993e8  0         5c2f03d0 a19be30     S         2   1c05a           0
```

Output description

address

Is the address of the lock in the lock table

If a user thread is waiting for this lock, the address of the lock shows in the **wait** field of the onstat -u (users) output.

wtlist

Is the first entry in the list of user threads that is waiting for the lock, if there is one

owner

Is the shared-memory address of the thread that is holding the lock

This address corresponds to the address in the **address** field of onstat -u (users) output. When the **owner** value is displayed in parentheses, it represents the shared memory address of a transaction structure. This scenario is possible only when a lock is allocated for a global transaction. This address corresponds to the address field of the output for onstat -G.

lklist

Is the next lock in a linked list of locks that are held by the owner listed

type

Uses the following codes to indicate the type of lock:

HDR

Header

B

Bytes

S

Shared

X

Exclusive

I

Intent

U

Update

IX

Intent-exclusive

IS

Intent-shared

SIX

Shared, intent-exclusive

tblsnum

Is the tblspace number of the locked resource. If the number is less than 10000, it indicates Enterprise Replication pseudo locks.

rowid

Is the row identification number

The rowid provides the following lock information:

- If the rowid equals zero, the lock is a table lock.
- If the rowid ends in two zeros, the lock is a page lock.
- If the rowid is six digits or fewer and does not end in zero, the lock is probably a row lock.
- If the rowid is more than six digits, the lock is probably an index key-value lock.

key#/bsiz

Is the index key number, or the number of bytes locked for a VARCHAR lock

If this field contains 'K-' followed by a value, it is a key lock. The value identifies which index is being locked. For example, K-1 indicates a lock on the first index that is defined for the table.

onstat -l command: Print physical and logical log information

Use the onstat -l command to display information about the physical logs, logical logs, and temporary logical logs.

Syntax:

```
onstat -l
```

Example Output

Figure 279. onstat -l command output

```
Physical Logging
Buffer bufused  bufsize  numpages numwrits pages/io
  P-1  0        16       716      55       13.02
      phybegin         physize  phypos   phyused  %used
      1:263            500      270      0        0.00

Logical Logging
Buffer bufused  bufsize  numrecs  numpages numwrits recs/pages pages/io
  L-3  0        16       42169    2872     1043     14.7       2.8
        Subsystem    numrecs  Log Space used
        OLDRSAM      42169    4436496

address  number   flags    uniqid   begin                size     used    %used
a517f70  1        U-B----  1        1:763                 500      500   100.00
a517fb0  2        U-B----  2        1:1263                500      500   100.00
a40daf0  3        U-B----  3        1:1763                500      500   100.00
a40db30  4        U-B----  4        1:2263                500      500   100.00
a40db70  5        U-B----  5        1:2763                500      500   100.00
a40dbb0  6        U---C-L  6        1:3263                500      372    74.40
a40dbf0  7        A------  0        1:3763                500        0     0.00
a40dc30  8        A------  0        1:4263                500        0     0.00
 8 active, 8 total
```

Output description for the physical log files

The first section of the display describes the physical-log configuration:

buffer

Is the number of the physical-log buffer

bufused

Is the number of pages of the physical-log buffer that are used

bufsize

Is the size of each physical-log buffer in pages

numpages

Is the number of pages written to the physical log

numwrits

Is the number of writes to disk

pages/io

Is calculated as `numpages/numwrits`

This value indicates how effectively physical-log writes are being buffered.

phybegin

Is the physical page number of the beginning of the log

physize

Is the size of the physical log in pages

phypos

Is the current position in the log where the next log-record write is to occur

phyused

Is the number of pages used in the log

%used

Is the percent of pages used

The second section of the onstat -l command output describes the logical-log configuration:

buffer

Is the number of the logical-log buffer

bufused

Is the number of pages used in the logical-log buffer

bufsize

Is the size of each logical-log buffer in pages

numrecs

Is the number of records written

numpages

Is the number of pages written

numwrits

Is the number of writes to the logical log

recs/pages

Is calculated as `numrecs/numpages`

You cannot affect this value. Different types of operations generate different types (and sizes) of records.

pages/io

is calculated as `numpages/numwrits`

You can affect this value by changing the size of the logical-log buffer (specified as LOGBUFF in the `ONCONFIG` file) or by changing the logging mode of the database (from buffered to unbuffered, or vice versa).

The following fields are repeated for each logical-log file:

address

Is the address of the log-file descriptor

number

Is logid number for the logical-log file

The logid numbers might be out of sequence because either the database server or administrator can insert a log file in-line.

flags

Provides the status of each log as follows:

A

Newly added (and ready to use)

B

Backed up

C

Current® logical-log file

D

Marked for deletion

To drop the log file and free its space for reuse, you must perform a level-0 backup of all storage spaces

F

Free, available for use

L

The most recent checkpoint record

U

Used

uniqid

Is the unique ID number of the log

begin

Is the beginning page of the log file

size

Is the size of the log in pages

used

Is the number of pages used

%used

Is the percent of pages used

active

Is the number of active logical logs

total

Is the total number of logical logs

Output description for temporary logical log files

The database server uses *temporary logical logs* during a warm restore because the permanent logs are not available then. The following fields are repeated for each temporary logical-log file:

address

Is the address of the log-file descriptor

number

Is logid number for the logical-log file

flags

Provides the status of each log as follows:

B

Backed up

C

Current® logical-log file

F

Free, available for use

U

Used

uniqid

Is the unique ID number of the log

begin

Is the beginning page of the log file

size

Is the size of the log in pages

used

Is the number of pages used

%used

Is the percent of pages used

active

Is the number of active temporary logical logs

onstat -L command: Print the number of free locks

Use the onstat -L command to print the number of free locks on a lock-free list.

Syntax:

```
onstat -L
```

Example output

Figure 281. onstat -L output

```
num   list head     available locks
0     10a143b70     19996
1     101010101     200
3     020202020     300
```

Output description

num

The list number

list head

The starting address of the list

available locks

The number of locks on this list

onstat -m command: Print recent system message log information

Use the onstat -m command to display the 20 most recent lines of the system message log.

You can use the onstat -m command option with the database server in any mode, including offline.

Syntax:

```
onstat -m
```

Example output

Output from this command lists the full pathname of the message log file and the 20 file entries. A date-and-time header separates the entries for each day. A time stamp prefaces single entries within each day. The name of the message log is specified as MSGPATH in the **ONCONFIG** file.

Figure 283. onstat -m command output

```
Message Log File: /work/11.50/dbspaces/star3.log
11:26:33  Checkpoint Completed:  duration was 0 seconds.
11:26:33  Checkpoint loguniq 1, logpos 0x23c408, timestamp: 0x2cc2 Interval: 9
```

onstat -o command: Output shared memory contents to a file

Use the onstat -o command to write the contents of shared memory to a specified file for later analysis. If you do not specify an output file, a file named **onstat.out** is created in the current directory.

Syntax:

onstat -o [{ nobuffs | full }] { *[outfile]* }

Use the **nobuffs** option to exclude the buffer pool in the resident segment of shared memory from the output file. This results in a smaller output file.

Use the **full** option to create an output file that is the same size as the shared memory segments for the HCL OneDB™ instance. You must have enough room in the file system to handle the output.

If you do not specify either the **nobuffs** or the **full** option, the output is controlled by the database server DUMPSHMEM configuration parameter setting:

- If DUMPSHMEM is set to 0 or to 1, onstat -o command writes a full shared-memory dump file.
- If DUMPSHMEM is set to 2, onstat -o command writes a **nobuffs** shared-memory dump file that excludes the buffer pool in the resident segment.

By running additional onstat commands against the file, you can gather information from a previously saved shared memory dump. The *outfile* that you create with the onstat -o command is the *infile* that you can use as a source file to run the additional onstat commands. For more information, see Running onstat Commands on a Shared Memory Dump File on page 484.

onstat -O command: Print optical subsystem information

Use the onstat -O command to display information about the Optical Subsystem memory cache and the staging area blobspace.

Syntax:

```
onstat -O
```

Example output

The totals shown in the display accumulate from session to session. The database server resets the totals to `0` only when you run the onstat -z command.

Figure 286. onstat -O command output

```
Optical StageBlob Cache
System Cache Totals:                                System Blob Totals:
Size     Alloc.   Avail.             Number   Kbytes  Number   Kbytes
500      500      0                  1        20      3        1500

User Cache Totals:                                  User Blob Totals:
SID      User     Size               Number   Kbytes  Number   Kbytes
94       doug     250                1        20      1        300
95       beth     500                0        0       2        1200
```

Output description

The first section of the display provides the following information on system-cache totals:

size

Is the size that the OPCACHEMAX configuration parameter specifies

alloc

Is the number of 1-kilobyte allocations to the cache

avail

Describes how much of **alloc** (in kilobytes) is not used

number

Is the number of simple large objects that the database server successfully put in the cache without overflowing

kbytes

Is the number of kilobytes of TEXT or BYTE data that the database server put in the cache without overflowing

number

Is the number of simple large objects that the database server wrote to the staging-area blobspace

kbytes

Is the number of kilobytes of TEXT or BYTE data that the database server wrote to the staging-area blobspace

Although the **size** output indicates the amount of memory that is specified in the configuration parameter OPCACHEMAX, the database server does not allocate memory to OPCACHEMAX until necessary. Therefore, the **alloc** output reflects only the number of 1-kilobyte allocations of the largest simple large object that has been processed. When the values in the **alloc** and **avail** output are equal to each other, the cache is empty.

The second section of the display describes the following user-cache totals information:

SID

Is the session ID for the user

user

Is the user ID of the client

size

Is the size specified in the **IONEDB_ OPCACHE** environment variable, if it is set

If you do not set the **IONEDB_ OPCACHE** environment variable, the database server uses the size that you specify in the configuration parameter OPCACHEMAX.

number

Is the number of simple large objects that the database server put into cache without overflowing

kbytes

Is the number of kilobytes of TEXT or BYTE data that the database server put in the cache without overflowing

number

Is the number of simple large objects that the database server wrote to the staging-area blobspace

kbytes

Is the number of kilobytes of TEXT or BYTE data that the database server wrote to the staging-area blobspace

The last line of the display lists the total number of sessions that are using the cache.

onstat -p command: Print profile counts

Use the onstat -p command to display information about profile counts either since you started the database server or since you ran the onstat -z command.

Syntax:

```
onstat -p
```

Example output

Figure 288. onstat -p command output

```
Profile
dskreads  pagreads   bufreads   %cached dskwrits   pagwrits   bufwrits   %cached
16934     47321      203600361  99.99   103113     158697     950932     89.16

isamtot   open     start    read     write   rewrite  delete   commit  rollbk
139214865 9195777  12257208 94191268 362691  55696    38134    128294  24

gp_read   gp_write   gp_rewrt   gp_del     gp_alloc   gp_free   gp_curs
39        2          27         51         0          0         16

ovlock    ovuserthread   ovbuff   usercpu  syscpu   numckpts   flushes
0         0              0        1551.59  144.82   1822       1822

bufwaits  lokwaits  lockreqs   deadlks  dltouts  ckpwaits  compress  seqscans
176       1         195872383  0        0        1         39331     1259170

ixda-RA   idx-RA     da-RA    logrec-RA  RA-pgsused lchwaits
0         7594       2124     0          2002       18848
```

Output description

The first portion of the output describes reads and writes.

Reads and writes are tabulated in three categories: from disk, from buffers, and number of pages (read or written).

The first **%cached** field is a measure of the number of reads from buffers compared to reads from disk. The second **%cached** field is a measure of the number of writes to buffers compared to writes to disk.

The database server buffers the information and writes the information to the disk in pages. For this reason, the number of disk writes displayed as `dskwrits` is usually less than the number of writes that an individual user runs:

dskreads

The number of actual reads from disk

pagreads

The number of pages read

bufreads

Is the number of reads from shared memory

%cached

The percent of reads cached in the buffer pool.

If `bufreads` exceeds the maximum integer (or long) value, its internal representation becomes a negative number, but the value appears as `0.0`.

dskwrits

The actual number of physical writes to disk

This number includes the writes for the physical and logical logs reported in onstat -l .

pagwrits

The number of pages written

bufwrits

The number of writes to shared memory

%cached

The percent of writes cached in the buffer pool.

The next portion of the **-p** display tabulates the number of times different ISAM calls were executed. The calls occur at the lowest level of operation and do not necessarily correspond one-to-one with SQL statement execution. A single query might generate multiple ISAM calls. These statistics are gathered across the database server and cannot be used to monitor activity on a single database unless only one database is active or only one database exists:

isamtot

The total number of calls

open

Increments when a tblspace is opened

start

Increments the pointer within an index

read

Increments when the read function is called

write

Increments with each write call

rewrite

Increments when an update occurs

delete

Increments when a row is deleted

commit

Increments each time that an **iscommit()** call is made

No one-to-one correspondence exists between this value and the number of explicit COMMIT WORK statements that are executed.

rollbk

Increments when a transaction is rolled back

The next portion of the onstat -p command output displays information about generic pages. The Generic Page Manager provides an API for OneDB to manage nonstandard pages in the database server buffer pool. The following table describes the Generic Page Manager fields in the onstat -p command output.

gp_read

The number of generic page reads

gp_write

The number of generic page writes

gp_rewrt

The number of generic page updates

gp_del

The number of generic page deletes

gp_alloc

The number of generic page allocations

gp_free

The number of generic pages freed and returned to tblspaces

gp_curs

The number of cursors used against generic pages

The next portion of the onstat -p command output displays the number of times that a resource was requested when none was available:

ovlock

Number of times that sessions attempted to exceed the maximum number of locks

For more information, see “LOCKS? on page 1-56.

ovuserthread

The number of times that a user attempted to exceed the maximum number of user threads

ovbuff

The number of times that the database server did not find a free shared-memory buffer

When no buffers are free, the database server writes a dirty buffer to disk and then tries to find a free buffer.

usercpu

Is the total user CPU time that all user threads use, expressed in seconds

This entry is updated every 15 seconds.

syscpu

The total system CPU time that all user threads use, expressed in seconds

This entry is updated every 15 seconds.

numckpts

The number of checkpoints since the boot time

flushes

The number of times that the buffer pool wasflushed to the disk

The next portion of the onstat -p command output contains miscellaneous information, as follows:

bufwaits

Increments each time that a user thread must wait for a buffer

lokwaits

Increments each time that a user thread must wait for a lock

lockreqs

Increments each time that a lock is requested

deadlks

Increments each time that a potential deadlock is detected and prevented

dltouts

Increments each time that the distributed deadlock time-out value is exceeded while a user thread is waiting for a lock

ckpwaits

Is the number of checkpoint waits

compress

Increments each time that a data page is compressed

seqscans

Increments for each sequential scan

*

The last portion of the onstat -p command output contains the following information:

ixda-RA

The count of read-aheads that go from index leaves to data pages

idx-RA

The count of read-aheads that traverse index leaves

da-RA

The count of data-path-only scans

logrec-RA

The log records that the database server read ahead

RA-pgsused

The number of pages used that the database server read ahead

lchwaits

Stores the number of times that a thread was required to wait for a shared-memory latch

Many latch waits typically results from a high volume of processing activity in which the database server is logging most of the transactions.

onstat -P command: Print partition information

Use the onstat -P command to display the partition number and the pages in the buffer pool for all of the partitions.

Syntax:

```
onstat  -P
```

For information about running onstat -P on a dump file created without the buffer pool, see Running onstat Commands on a Shared Memory Dump File on page 484.

Example output

Figure 290. onstat -P command output

```
Buffer pool page size: 2048
partnum  total    btree    data     other    dirty
0        36       1        8        27       0
1048577  2        0        0        2        0
1048578  4        1        1        2        0
1048579  23       10       12       1        0
1048580  68       31       36       1        0
4194309  3        0        1        2        0

Totals:  3000     786      1779     435      0
Percentages:
Data  59.30
Btree 26.20
Other 14.50

Buffer pool page size: 8192
partnum  total    btree    data     other    dirty
0        999      0        0        999      0
5242881  1        0        0        1        0

Totals:  1000     0        0        1000     0
Percentages:
Data  0.00
Btree 0.00
Other 100.00
```

Output description

Buffer pool page size

The size, in bytes, of the buffer pool pages.

partnum

The partition number.

total

The total number of partitions.

btree

The number of B-tree pages in the partition.

data

The number of data pages in the partition.

other

The number of other pages in the partition.

dirty

The number of dirty pages in the partition.

onstat -r command: Repeatedly print selected statistics

Use the onstat -r command to repeatedly print the statistics for other options specified in the command at specified intervals.

Syntax:

`onstat -r` [{ *secondsother_options* | *other_options* }]

Use the onstat -r ***seconds other_options*** command to specify the seconds to repeat the other option.

Use onstat -r ***other_options*** command to have the option repeat every five seconds, which allows the other options to be concatenated with the -r option, as in this example: `onstat -rFh`.

The onstat -r command can be used in both command mode and interactive mode, and can be useful for repeating command output to monitor system resource utilization.

Example output running the onstat -r command every five seconds

Figure 292. command output

```
onstat -r

IBM Informix Dynamic Server Version 11.70.F       -- On-Line -- Up 20:05:25 -- 1067288 Kbytes


IBM Informix Dynamic Server Version 11.70.F       -- On-Line -- Up 20:05:30 -- 1067288 Kbytes


IBM Informix Dynamic Server Version 11.70.F       -- On-Line -- Up 20:05:35 -- 1067288 Kbytes
```

Example output running the onstat -r command every ten seconds

Figure 293. command output

```
onstat -r 10

IBM Informix Dynamic Server Version 11.50.F       -- On-Line -- Up 20:06:58 -- 1067288 Kbytes


IBM Informix Dynamic Server Version 11.50.F       -- On-Line -- Up 20:07:08 -- 1067288 Kbytes


IBM Informix Dynamic Server Version 11.50.F       -- On-Line -- Up 20:07:18 -- 1067288 Kbytes
```

Example output running the onstat -r every one second, with the -h option

Figure 294. onstat -r 1 -h command output

```
onstat -r 1 -h

Buffer pool page size: 2048

buffer hash chain length histogram
   # of chains         of len
          3841              0
          3767              1
           522              2
            62              3
          8192   total chains
          4351   hashed buffs
          5000    total buffs


Buffer pool page size: 2048

buffer hash chain length histogram
   # of chains         of len
          4020              0
          3392              1
           735              2
            43              3
             2              4
          8192   total chains
          4172   hashed buffs
          5000    total buffs
```

Example output running the onstat -r command every five seconds, with the -Fh options

Figure 295. onstat -rFh command output

```
onstat -rFh


Fg Writes     LRU Writes    Chunk Writes
0             0             21

address           flusher  state    data     # LRU    Chunk    Wakeups  Idle Tim
460e6820         0        I        0        0        2        5        9.820
      states: Exit Idle Chunk Lru

Buffer pool page size: 2048

buffer hash chain length histogram
   # of chains         of len
          6342              0
          1850              1
          8192   total chains
          1850   hashed buffs
          5000    total buffs


Fg Writes     LRU Writes    Chunk Writes
0             0             21

address           flusher  state    data     # LRU    Chunk    Wakeups  Idle Tim
460e6820         0        I        0        0        2        10       22.755
      states: Exit Idle Chunk Lru

Buffer pool page size: 2048

buffer hash chain length histogram
   # of chains         of len
          4396              0
          3796              1
          8192   total chains
          3796   hashed buffs
          5000    total buffs
```

onstat -R command: Print LRU, FLRU, and MLRU queue information

Use the onstat -R command to display detailed information about the LRU queues, FLRU queues, and MLRU queues. For each queue, the onstat -R command displays the number of buffers in the queue and the number and percentage of buffers that have been modified.

For an in-depth discussion of the three types of queues, see LRU queues in the shared-memory chapter of the *HCL OneDB™ Administrator's Guide*.

Syntax:

```
onstat -R
```

Example output

Figure 297. onstat -R command output

```
    Buffer pool page size: 2048
    8 buffer LRU queue pairs              priority levels
# f/m   pair total     % of    length       LOW      HIGH
 0 f        375     100.0%      375        375          0
 1 m                  0.0%        0          0          0
 2 f        375     100.0%      375        375          0
 3 m                  0.0%        0          0          0
 4 f        375     100.0%      375        375          0
 5 m                  0.0%        0          0          0
 6 F        375     100.0%      375        375          0
 7 m                  0.0%        0          0          0
 8 f        375     100.0%      375        375          0
 9 m                  0.0%        0          0          0
10 f        375     100.0%      375        375          0
11 m                  0.0%        0          0          0
12 f        375     100.0%      375        375          0
13 m                  0.0%        0          0          0
14 f        375     100.0%      375        375          0
15 m                  0.0%        0          0          0
0 dirty, 3000 queued, 3000 total, 4096 hash buckets, 2048 buffer size
start clean at  60.000% (of pair total) dirty, or 226 buffs dirty, stop at
  50.000%
Buffer pool page size: 8192
    4 buffer LRU queue pairs              priority levels
# f/m   pair total     % of    length       LOW      HIGH
 0 F        250     100.0%      250        250          0
 1 m                  0.0%        0          0          0
 2 f        250     100.0%      250        250          0
 3 m                  0.0%        0          0          0
 4 f        250     100.0%      250        250          0
 5 m                  0.0%        0          0          0
 6 f        250     100.0%      250        250          0
 7 m                  0.0%        0          0          0
0 dirty, 1000 queued, 1000 total, 1024 hash buckets, 8192 buffer size
start clean at  60.000% (of pair total) dirty, or 150 buffs dirty, stop at
  50.000%
```

Output description

Buffer pool page size

Is the page size of the buffer pool in bytes

#

Shows the queue number

Each LRU queue is composed of two subqueues: an FLRU queue and a MLRU queue. (For a definition of FLRU and MLRU queues, see LRU queues in the shared-memory chapter of the *HCL OneDB™ Administrator's Guide*.) Thus, queues 0 and 1 belong to the first LRU queue, queues 2 and 3 belong to the second LRU queue, and so on.

f/m

Identifies queue type

This field has four possible values:

f

Free LRU queue

In this context, free means not modified. Although nearly all the buffers in an LRU queue are available for use, the database server attempts to use buffers from the FLRU queue rather than the MLRU queue. (A modified buffer must be written to disk before the database server can use the buffer.)

F

Free LRU with fewest elements

The database server uses this estimate to determine where to put unmodified (free) buffers next.

m

MLRU queue

M

MLRU queue that a flusher is cleaning

length

Tracks the length of the queue measured in buffers

% of

Shows the percent of LRU queue that this subqueue composes

For example, suppose that an LRU queue has 50 buffers, with 30 of those buffers in the MLRU queue and 20 in the FLRU queue. The **% of** column would list percents of 60.00 and 40.00, respectively.

pair total

Provides the total number of buffers in this LRU queue

priority levels

Displays the priority levels: `LOW, MED_LOW, MED_HIGH, HIGH`

The onstat -R command also lists the priority levels.

Summary information follows the individual LRU queue information. You can interpret the summary information as follows:

dirty

Is the total number of buffers that have been modified in all LRU queues

queued

Is the total number of buffers in LRU queues

total

Is the total number of buffers

hash buckets

Is the number of hash buckets

buffer size

Is the size of each buffer

start clean

Is the value specified in the **lru_max_dirty** field of the BUFFERPOOL configuration parameter

stop at

Is the value specified in the **lru_min_dirty** field of the BUFFERPOOL configuration parameter

priority downgrades

Is the number of LRU queues downgraded to a lower priority.

priority upgrades

Is the number of LRU queues upgraded to a higher priority.

onstat -s command: Print latch information

Use the onstat -s command to display general latch information, including the resource that the latch controls.

Syntax:

```
onstat  -s
```

Example output

Figure 299. onstat -s command output

```
OneDB Version 1.0.1.0         -- On-Line (CKPT REQ)
-- Up 00:19:21 -- 167608 Kbytes
Blocked:CKPT

Latches with lock or userthread set
name     address          lock wait userthread
physlog  4410d5a8         0    0    45a616b8
```

Output description

name

Identifies the resource that the latch controls with the following abbreviations:

archive

Storage-space backup

bf

Buffers

bh

Hash buffers

chunks

Chunk table

ckpt

Checkpoints

dbspace

Dbspace table

flushctl

Page-flusher control

flushr

Page cleaners

locks

Lock table

loglog

Logical log

LRU

LRU queues

physb1

First physical-log buffer

physb2

Second physical-log buffer

physlog

Physical log

pt

Tblspace tblspace

tblsps

Tblspace table

users

User table

address

Is the address of the latch

This address appears in the onstat -u (users) command output wait field if a thread is waiting for the latch.

lock

Indicates if the latch is locked and set

The codes that indicate the lock status (`1` or `0`) are computer dependent.

wait

Is the shared-memory address of the user thread that is waiting for a latch, or is blank when no user threads are waiting

userthread

Is the shared-memory address of the user thread that holds the latch

This address corresponds to the value in the **tcb** column of the onstat -g ath output. You can compare this address with the user addresses in the onstat -u output to obtain the user-process identification number.

onstat -t and onstat -T commands: Print tblspace information

Use the onstat -t command to display tblspace information for active tblspaces. Use the onstat -T command to display tblspace information for all tblspaces.

The onstat -t command also lists the number of active tblspaces and the total number of tblspaces.

Syntax:

```
onstat { -t | -T }
```

Example output

Figure 301. onstat -t command output

```
Tblspaces
 n address  flgs ucnt tblnum   physaddr         npages nused  npdata nrows  nextns
62 a40dc70  0    1    100001   1:14             250    250    0      0      1
195 ac843e0  0    1    1000df   1:236            16     9      4      53     2
 2 active, 221 total
```

Output description

n

Is a counter of open tblspaces

address

Is the address of the tblspace in the shared-memory tblspace table

flgs

Uses the following flag bits to describe the flag:

0x00000002

Flush the partition info at the next checkpoint.

0x00000004

Drop partition is in progress.

0x00000008

Partition is a pseudo partition (sysmaster).

0x00000020

ALTER TABLE is in progress.

0x00000040

Partition has been dropped.

0x00000800

Partition is the system temp table.

0x00001000

Partition is the user temp table.

0x00008000

Online index create or drop in progress.

0x00400000

A single user access to the partition is requested.

0x00800000

Drop partition is completed.

0x40000000

Flush the partition info. The partition flush can be delayed until later in the checkpoint process.

ucnt

Is the usage count, which indicates the number of user threads currently accessing the tblspace

tblnum

Is the tblspace number expressed as a hexadecimal value

The integer equivalent appears as the **partnum** value in the **systables** system catalog table.

physaddr

Is the physical address (on disk) of the tblspace

npages

Is the number of pages allocated to the tblspace

nused

Is the number of used pages in the tblspace

npdata

Is the number of data pages used

nrows

Is the number of data rows used

nextns

Is the number of noncontiguous extents allocated

This number is not the same as the number of times that a next extent has been allocated.

onstat -u command: Print user activity profile

Use the onstat -u command to display a profile of user activity.

Syntax:

```
onstat  -u
```

Example output

Figure 303. onstat -u command output

```
Userthreads
address  flags   sessid   user     tty      wait     tout locks nreads   nwrites
a4d8018  ---P--D 1        informix -        0        0    0     58       4595
a4d8628  ---P--F 0        informix -        0        0    0     0        2734
a4d8c38  ---P--- 5        informix -        0        0    0     0        1
a4d9248  ---P--B 6        informix -        0        0    0     40       0
a4d9858  ---P--D 7        informix -        0        0    0     0        0
a4d9e68  Y--P--- 21       niraj    -        a65e5a8  0    1     0        0
 6 active, 128 total, 7 maximum concurrent
```

Output description

The -u option displays the following output for each user thread.

address

The shared-memory address of the user thread in the user table.

Compare this address with the addresses displayed in the output from the -s option (latches); the output from the -b, -B, and -X options (buffers); and the output from the -k option (locks) to learn what resources this thread is holding or waiting for.

flags

Provides the status of the session.

The flag codes for position 1:

B

Waiting for a buffer

C

Waiting for a checkpoint

G

Waiting for a write of the logical-log buffer

L

Waiting for a lock

S

Waiting for mutex

T

Waiting for a transaction

Y

Waiting for condition

X

Waiting for a transaction cleanup (rollback)

DEFUNCT

The thread has incurred a serious assertion failure, and has been suspended to allow other threads to continue their work.

The flag code for position 2:

*****

Transaction active during an I/O failure

The flag code for position 3:

A

A dbspace backup thread

For other values that appear here, see the third position of flag codes for the -x option.

The flag code for position 4:

P

Primary thread for a session

The flag codes for position 5:

R

Reading

X

Thread in critical section

The flag codes for position 6:

R

Thread used in recovery (for example, physical or logical recovery)

-

Thread not used in recovery

The flag codes for position 7:

B

A B-tree cleaner thread

C

Terminated user thread waiting for cleanup

D

A daemon thread

F

A page-cleaner thread

M

Special ON-Monitor thread (UNIX™)

sessid

The session identification number.

During operations such as parallel sorting and parallel index building, a session might have many user threads associated with it. For this reason, the session ID identifies each unique session.

user

The user login name, which is derived from the operating system

tty

The name of the standard error (stderr) file that the user is using, which is derived from the operating system.

This field is blank on Windows™.

wait

If the user thread is waiting for a specific latch, lock, mutex, or condition, this field displays the address of the resource. Use this address to map to information provided in the output from the -s (latch) option or the -k (lock) option. If the wait is for a persistent condition, run a **grep** for the address in the output from the onstat -a command.

tout

The number of seconds left in the current wait

If the value is `0`, the user thread is not waiting for a latch or lock. If the value is `-1`, the user thread is in an indefinite wait.

locks

The number of locks that the user thread is holding

The -k output should include a listing for each lock held.)

nreads

The number of disk reads that the user thread has executed

nwrites

The number of write calls that the user thread has executed

All write calls are writes to the shared-memory buffer cache.

The last line of the onstat -u command output displays the maximum number of concurrent user threads that were allocated since you initialized the database server. For example, the last line of a sample onstat -u command output is as follows:

```
4 active, 128 total, 17 maximum concurrent
```

The last part of the line, `17 maximum concurrent`, indicates that the maximum number of user threads that were running concurrently since you initialized the database server is 17.

The output also indicates the number of active users and the maximum number of users allowed.

onstat -x command: Print database server transaction information

Use the onstat -x command to display transaction information on the database server.

Syntax:

```
onstat -x
```

The transaction information is required only in the following situations:

- X/Open environment
- Database server participation in distributed queries
- Database server uses the Microsoft™ Transaction Server (MTS) transaction manager

Example output

Figure 305. onstat -x command output

```
Transactions

address   flags userthread  locks  begin    current     isol   est.     retrys coord
                                    logpos   logpos             rb_time
a6d8028   A---- a695028     0      -         -          COMMIT  -        0
a6d8348   A---- a695878     0      -         -          COMMIT  -        0
a6d8668   A---- a6960c8     0      -         -          COMMIT  -        0
a6d8988   A---- a696918     0      -         -          COMMIT  -        0
a6d8fc8   A---- a698208     0      -         -          COMMIT  -        0
a6d92e8   A---- a6979b8     0      -         -          COMMIT  -        0
a6d9608   A---- a698a58     0      -         -          COMMIT  -        0
a6d9928   A---- a6992a8     1      -         -          DIRTY   -        0
a6d9c48   A---- a6992a8     0      -         -          NOTRANS -        0
a6d9f68   A---- a69a348     0      -         -          COMMIT  -        0
a6da288   A---- a69ab98     0      -         -          COMMIT  -        0
a6da5a8   A---- a69b3e8     0      -         -          COMMIT  -        0
a6da8c8   A---- a69bc38     0      -         -          COMMIT  -        0
a6dabe8   A---- a69c488     0      -         -          COMMIT  -        0
a6daf08   A---- a699af8     0      -         -          COMMIT  -        0
a6db228   A---- a6992a8     0      -         -          COMMIT  -        0
a6db548   A---- a69ccd8     1      -         -          DIRTY   -        0
a6db868   A---- a69d528     1      -         -          DIRTY   -        0
a6dbb88   A---- a69ccd8     0      -         -          COMMIT  -        0
a6dbea8   A---- a69dd78     0      -         -          COMMIT  -        0
a6dc1c8   A---- a69e5c8     0      -         -          COMMIT  -        0
a6dc4e8   A-B-- a69ee18   502    33:0x25018  34:0x486fc COMMIT  0:07     0
 22 active, 128 total, 23 maximum concurrent
```

Output description

You can interpret output from the onstat -x command as follows:

address

The shared-memory address of the transaction structure

flags

The flag codes for position 1 (current transaction state):

A

User thread attached to the transaction

S

TP/XA suspended transaction

C

TP/XA waiting for rollback

The flag codes for position 2 (transaction mode):

T

Tightly-coupled mode (MTS)

L

Loosely-coupled mode (default mode)

The flag codes for position 3 (transaction stage):

B

Begin work

P

Distributed query prepared for commit

X

TP/XA prepared for commit

C

Committing or committed

R

Rolling back or rolled back

H

Heuristically rolling back or rolled back

The flag code for position 4:

X

XA transaction

The flag codes for position 5 (type of transaction):

G

Global transaction

C

Distributed query coordinator

S

Distributed query subordinate

B

Both distributed query coordinator and subordinate

M

Redirected global transaction

userthread

The thread that owns the transaction (**rstcb** address)

locks

The number of locks that the transaction holds

begin logpos

The position within the log when the BEGIN WORK record was logged.

current logpos

The current log position of the most recent record that the transaction is wrote too (As a transaction rolls back, the current log position will actually wind back until it gets to the beginning log position. When the beginning log position is reached, the rollback is complete.)

isol

The isolation level.

est. rb time

The estimated time the server needs to rollback the transaction. As a transaction goes forward, this time increases. If a transaction rolls back, the time decreases as the transaction unwinds.

retrys

Are the attempts to start a recovery thread for the distributed query

coord

The name of the transaction coordinator when the subordinate is executing the transaction

This field tells you which database server is coordinating the two-phase commit.

The last line of the onstat -x command output indicates that `8` is the maximum number of concurrent transactions since you initialized the database server.

```
8 active, 128 total, 8 maximum concurrent
```

Determine the position of a logical-log record

Use the onstat -x command to determine the position of a logical-log record.

The **curlog** and **logposit** fields provide the exact position of a logical-log record. If a transaction is not rolling back, **curlog** and **logposit** describe the position of the most recently written log record. When a transaction is rolling back, these fields describe the position of the most recently undone log record. As the transaction rolls back, the **curlog** and **logposit** values decrease. In a long transaction, the rate at which the **logposit** and **beginlg** values converge can help you estimate how much longer the rollback is going to take.

For an onstat -x command example, see monitoring a global transaction in the chapter on multiphase commit protocols in the *HCL OneDB™ Administrator's Guide*.

Determine the mode of a global transaction

The onstat -x command is useful for determining whether a global transaction is executing in loosely-coupled or tightly-coupled mode.

The second position of the flags column in the output from the onstat -x command displays the flags for global transactions. The `T` flag indicates tightly-coupled mode and the `L` flag indicates loosely-coupled mode.

- *Loosely-coupled mode* means that the different database servers coordinate transactions but do not share locks. Each branch in a global transaction has a separate transaction XID. The records from all branches display as separate transactions in the logical log.
- *Tightly-coupled mode* means that the different database servers coordinate transactions and share resources such as locking and logging. In a global transaction, all branches that access the same database share the same transaction XID. Log records for branches with the same XID appear under the same session ID. MTS uses tightly-coupled mode.

onstat -X command: Print thread information

Use the onstat -X command to obtain precise information about the threads that are waiting for buffers.

For each buffer in use, the onstat -X command displays general buffer information that is also available with either the onstat -b or onstat -B commands. For more information, refer to the onstat -b command in onstat -b command: Print buffer information for buffers in use on page 485.

Syntax:

```
onstat -X
```

Example output

Figure 307. onstat -X command output

```
    Buffers (Access)
address  owner    flags pagenum          memaddr  nslots pgflgs scount   waiter
    Buffer pool page size: 2048
 0 modified, 3000 total, 4096 hash buckets, 2048 buffer size
    Buffer pool page size: 8192
 0 modified, 1000 total, 1024 hash buckets, 8192 buffer size
```

Output description

The onstat -X command has a **waiter** field to list all user threads that are waiting for the buffer, whereas the onstat -b and onstat -B commands contain a **waitlist** field that displays the address of the first user thread that is waiting for the buffer. The maximum number of shared buffers is specified in the **buffers** field in the BUFFERPOOL configuration parameter in the `ONCONFIG` file.

Buffer pool page size

The size of the buffer pool pages in bytes

address

The address of the buffer header in the buffer table

flags

Flags identifying the current status of the buffer page:

0x01

Modified Data

0x02

Data

0x04

LRU

0x08

Error

0x10

Shared lock

0x20

LRU AIO write in progress

0x40

Chunk write in progress

0x80

Exclusive lock

0x100

Cleaner assigned to LRU

0x200

Buffer should avoid bf_check calls

0x400

Do log flush before writing page

0x800

Buffer has been 'buff' -checked

0x8000

Buffer has been pinned

pagenum

The physical page number on the disk

memaddr

The buffer memory address

nslots

The number of slot-table entries in the page

This field indicates the number of rows (or portions of a row) that are stored on the page.

pgflgs

Uses the following values, alone or in combination, to describe the page type:

1

Data page

2

Tblspace page

4

Free-list page

8

Chunk free-list page

9

Remainder data page

b

Partition resident blobpage

c

Blobspace resident blobpage

d

Blob chunk free-list bit page

e

Blob chunk blob map page

10

B-tree node page

20

B-tree root-node page

40

B-tree branch-node page

80

B-tree leaf-node page

100

Logical-log page

200

Last page of logical log

400

Sync page of logical log

800

Physical log

1000

Reserved root page

2000

No physical log required

8000

B-tree leaf with default flags

scount

Displays the number of threads that are waiting for the buffer

waiter

Lists the addresses of all user threads that are waiting for the buffer

onstat -z command: Clear statistics

Use the onstat -z command to clear database server statistics, including statistics that relate to Enterprise Replication, and set the profile counts to `0`.

If you use the onstat -z command to reset and monitor the count of some fields, be aware that profile counts are incremented for all activity that occurs in any database that the database server manages. Any user can reset the profile counts and thus interfere with monitoring that another user is conducting.

Syntax:

```
onstat  -z
```

Return codes on exiting the onstat utility

The onstat utility displays a set of return codes when you exit the utility.

Example

The following lines are an example of the messages and return codes that are displayed when you exit the onstat utility:

```
GLS failures: -1
Failed to attach shared memory: -1
Failed to attach shared memory when running 'onstat -': 255
All other errors detected by onstat: 1
No errors detected by onstat: 0
Administration mode: 7
```

Return code values

The following table lists the database server mode that corresponds to the return codes that are displayed when you exit the onstat utility.

Value	Explanation
-1	GLS locale initialization failed or HCL OneDB™ failed to attach to shared memory
0	Initialization mode
1	Quiescent mode
2	Recovery mode
3	Backup mode
4	Shutdown mode
5	Online mode
6	Abort mode
7	User mode
255	Off-Line mode

SQL Administration API

SQL Administration API Functions

These topics describe the SQL administration API admin() and task() functions.

SQL Administration API Overview

Use the SQL administration API to remotely administer HCL OneDB™ through SQL statements.

The SQL administration API consists of two functions: admin() and task(). These functions perform the same operations, but return results in different formats. These functions take one or more arguments that define the operation. Many of the operations are ones that you can also complete with command line utilities. The advantage of using the SQL administration API functions is that you can run them remotely from other database servers; whereas you must be directly connected to the database server on which to run command line utility commands.

You can invoke the admin() and task() functions within SQL statements that can include an expression, or you can use the EXECUTE FUNCTION statement to call them. Run the admin() or task() function within a transaction that does not include any other statements.

Note: When connected to a secondary node, the admin() function is disabled and will always return a value of -1. In addition, the task() function will return an error for commands that involve modifying disk structures, since these administrative actions are meant to be executed only on primary or standalone nodes.

The SQL administration API functions are defined in the **sysadmin** database. You must be connected to the **sysadmin** database, either directly or remotely, to run these functions.

The SQL administration API functions can be run only by the following users:

- The user **informix**
- The **root** user, if Connect privilege on the **sysadmin** database is granted to the user
- The DBSA group members, if Connect privilege on the **sysadmin** database is granted to the role
- Users granted privileges to SQL administration API commands by the admin() and task() functions with grant admin argument.

You can generate SQL administration API commands for reproducing the storage spaces, chunks, and logs that exist in a file. To do this, run the **dbschema** utility with the **-c** option.

admin() and task() Function Syntax Behavior

The admin() and task() functions take one or more arguments as quoted strings separated by commas.

The syntax for the admin() and task() functions includes the following rules:

- Each argument must be delimited by a pair of single (') quotation marks or double (") quotation marks.
- Arguments must be separated by a comma.
- The maximum number of arguments is 28.
- Most arguments are not case-sensitive, with the following exceptions:
    - The argument that immediately follows the initial **onmode** argument is case-sensitive.

      For example:

      ```
      EXECUTE FUNCTION task("onmode","D","50");
      ```
    - The arguments included with the **cdr** argument are case-sensitive.

For example:

```
EXECUTE FUNCTION task("cdr define server",
   "-c=g_amsterdam","--init g_amsterdam");
```

- If you are not directly connected to the **sysadmin** database, you must include the **sysadmin** database name and the server name, according to the standard Database Object Name syntax. For example, if your server name is **ids_server**, you could run the following statement:

```
EXECUTE FUNCTION sysadmin@ids_server:admin("add bufferpool","2",
"50000","8","60.0","50.0");
```

  For more information on the Database Object Name syntax, see the *HCL OneDB™ Guide to SQL: Syntax*.

admin() and task() Argument Size Specifications

By default, the units for arguments specifying sizes in admin() and task() functions are kilobytes. You can specify other units.

You can use the following units in size arguments to the admin() and task() functions:

Notation

Corresponding Units

`B`

Bytes

`K`

Kilobytes (Default)

`M`

Megabytes

`G`

Gigabytes

`T`

Terabytes

`P`

Petabytes

The letter case of these characters is ignored.

Any white space that separates the size specification and the units abbreviation in the same argument is ignored. For example, the specifications `"128M"` and `"128 m"` are both interpreted as 128 megabytes.

When a size argument is omitted, the default size for that object applies, based either on the setting of a configuration parameter, or on the system default if no parameter is set. Storage spaces, for example, have a default size of 100 megabytes.

admin() and task() Function Return Codes

The admin() and task() functions perform equivalent tasks but produce different types of return codes. Use the admin() function if you want an integer return code, or the task() function if you want a textual return code.

When you run the admin() or task() function, it:

- Performs the specified operation.
- Returns a value that signifies whether the function succeeded or failed.
- Inserts a row into the **command_history** table of the **sysadmin** database.

The return codes for the admin() and task() functions indicate whether the function succeeded or failed in different formats:

- The task() function returns a textual message. The message is also inserted into the **cmd_ret_msg** column in the new row that the task() function inserts into the **command_history** table.
- The admin() function returns an integer. This number is also inserted into the **cmd_number** column in the new row that the admin() function inserts into the **command_history** table.
    - If this value is greater than zero, the function succeeded, and a new row was inserted into the **command_history** table.
    - If this value is zero, the function succeeded, but HCL OneDB™ could not insert a new row into the **command_history** table.
    - If this value is less than zero, the function failed, but a new row was inserted into the **command_history** table.

The operation that the admin() or task() function specifies occurs in a separate transaction from the insertion of the new row into the **command_history** table. If the command executes successfully, but the insertion into the **command_history** table fails, the command takes effect, but an **online.log** error entry indicates the problem.

If the **command_history.cmd_number** serial counter is 200 when this function is called, and the command succeeds, then HCL OneDB™ executes the command and returns the integer `201`. If the command fails, this example returns the value `-201`.

Suppose the task() function had executed the same command:

```
EXECUTE FUNCTION task("check extents");
```

This command instructs the database server to check the extents, and returns a message indicating whether the command succeeded or failed.

If the **command_history.cmd_number** serial counter is 201 when this function is called, and the command fails, then the returned value is `-202`. Suppose that the next SQL administration API function that the DBSA invokes is this:

```
EXECUTE FUNCTION admin('create dbspace',
   'dbspace2', '/work/CHUNKS/dbspace2', 20M);
```

If in this case the command succeeds, the returned value is `203`. The DBSA can use the following query to examine the two rows of the **command_history** table that these calls to the admin() function inserted:

```
SELECT * FROM command_history WHERE cmd_number IN (202,203);
```

This query returns two rows:

```
cmd_number      202
cmd_exec_time   2009-04-17 16:26:14
cmd_user        informix
cmd_hostname    olympia
cmd_executed    create dbspace
cmd_ret_status  -1
cmd_ret_msg     Unable to create file /work/dbspace2

cmd_number      203
cmd_exec_time   2009-04-17 16:26:15
cmd_user        informix
cmd_hostname    olympia
cmd_executed    create dbspace
cmd_ret_status  0
cmd_ret_msg     created dbspace number 2 named dbspace2
```

Note: When connected to a secondary node, the admin() function is disabled and will always return a value of -1. In addition, the task() function will return an error for commands that involve modifying disk structures, since these administrative actions are meant to be executed only on primary or standalone nodes.

SQL administration API portal: Arguments by functional category

You can view a list of admin() and task() function arguments, sorted by category, with links to information about the arguments.

Category list

To use this section, you first determine the appropriate category from the following list, then follow the link to the SQL administration API function arguments for that category.

- Compression arguments on page 721
- Configuration parameter arguments on page 721
- Data, partition, and extent arguments on page 722
- Database arguments on page 722
- Enterprise replication arguments on page 723
- High availability arguments on page 723
- Listen thread arguments on page 724
- Log arguments on page 724
- Message log arguments on page 725
- Memory arguments on page 725
- Mirror arguments on page 726
- Parallel database query (PDQ) arguments on page 726
- Server mode arguments on page 726
- Space arguments on page 727
- Storage provisioning arguments on page 728

- SQL tracing arguments on page 730
- Miscellaneous arguments on page 730

Backup arguments

Use the following SQL administration API function arguments to backup your databases.

Table 159. admin() and task() Function Arguments for Databases

Argument	When Added to OneDB
ontape archive argument: Backup the data on your database (SQL administration API)	Version 11.70.xC2
onbar argument: Backup the storage spaces (SQL administration API)	Version 11.70.xC2
onsmsync argument: Synchronize with the storage manager catalog (SQL administration API)	Version 11.70.xC2

Compression arguments

Use the following SQL administration API function arguments to manage the compression of data and to optimize storage.

Table 160. admin() and task() Function Arguments for Compression Commands

Argument	When Added to OneDB
index compress repack shrink arguments: Optimize the storage of B-tree indexes (SQL administration API)	Version 12.10
index estimate_compression argument: Estimate index compression (SQL administration API)	Version 12.10
table or fragment arguments: Compress data and optimize storage (SQL administration API)	Version 11.50xC4
table or fragment arguments: Compress data and optimize storage (SQL administration API)	Version 11.50xC4
purge compression dictionary arguments: Remove compression dictionaries (SQL administration API)	Version 11.50xC4

For an overview of compression and storage optimization commands, see Table and fragment compress and uncompress operations (SQL administration API).

Configuration parameter arguments

Use the following SQL administration API function arguments to update configuration parameters.

Table 161. admin() and task() Function Arguments for Configuration Parameter Commands

Argument	When Added to OneDB
export config argument: Export configuration parameter values (SQL administration API)	Version 12.10.xC1
import config argument: Import configuration parameter values (SQL administration API)	Version 12.10.xC1
modify config arguments: Modify configuration parameters (SQL administration API)	Version 12.10.xC1
onmode and wf arguments: Permanently update a configuration parameter (SQL administration API)	Version 11.10.xC1
onmode and wm arguments: Temporarily update a configuration parameter (SQL administration API)	Version 11.10.xC1
onmode, wm, and AUTO_LRU_TUNING arguments: Change LRU tuning status (SQL administration API)	Version 11.10.xC1
reset config argument: Revert configuration parameter value (SQL administration API)	Version 12.10.xC1
reset config all argument: Revert all dynamically updatable configuration parameter values (SQL administration API)	Version 12.10.xC1
set onconfig permanent argument: Permanently change a configuration parameter (SQL administration API)	Version 11.50.xC3
set onconfig memory argument: Temporarily change a configuration parameter (SQL administration API)	Version 11.50.xC3

Database arguments

Use the following SQL administration API function arguments to create and drop databases.

Table 162. admin() and task() Function Arguments for Databases

Argument	When Added to OneDB
create database argument: Create a database (SQL administration API)	Version 11.70.xC2
drop database argument: Drop a database (SQL administration API)	Version 11.70.xC2

Data, partition, and extent arguments

Use the following SQL administration API function arguments to manage data, partitions, and extents.

Table 163. admin() and task() Function Arguments for Data, Partition, and Extent Commands

Argument	When Added to OneDB
onmode and C arguments: Control the B-tree scanner (SQL administration API)	Version 11.10.xC1
onmode and c arguments: Force a checkpoint (SQL administration API)	Version 11.10.xC1
check data argument: Check data consistency (SQL administration API)	Version 11.10.xC1
check extents argument: Check extent consistency (SQL administration API)	Version 11.10.xC1
check partition argument: Check partition consistency (SQL administration API)	Version 11.10.xC1
checkpoint argument: Force a checkpoint (SQL administration API)	Version 11.10.xC1
print partition argument: Print partition information (SQL administration API)	Version 11.10.xC1
set dataskip argument: Start or stop skipping a dbspace (SQL administration API)	Version 11.10.xC1
set index compression argument: Change index page compression (SQL administration API)	Version 11.50.xC2

Enterprise replication arguments

Use the following SQL administration API function arguments to manage Enterprise Replication.

Table 164. admin() and task() Function Arguments for Enterprise Replication Commands

Argument	When Added to OneDB
cdr argument: Administer Enterprise Replication (SQL administration API)	Version 11.50.xC3

High availability arguments

Use the following SQL administration API function arguments to manage high-availability replication (HDR).

Table 165. admin() and task() Function Arguments for HDR Commands

Argument	When Added to OneDB
archive fake argument: Perform an unrecorded backup (SQL administration API) on page 752	Version 11.50.xC1
ha make primary argument: Change the mode of a secondary server (SQL administration API)	Version 11.50.xC1
ha rss argument: Create an RS secondary server (SQL administration API)	Version 11.50.xC1
ha rss add argument: Add an RS secondary server to a primary server (SQL administration API)	Version 11.50.xC1
ha rss change argument: Change the password of an RS secondary server (SQL administration API)	Version 11.50.xC1

Table 165. admin() and task() Function Arguments for HDR Commands (continued)

Argument	When Added to OneDB
ha rss delete argument: Delete an RS secondary server (SQL administration API)	Version 11.50.xC1
ha sds clear argument: Stop shared-disk replication (SQL administration API)	Version 11.50.xC1
ha sds primary argument: Convert an SD secondary server to a primary server (SQL administration API)	Version 11.50.xC1
ha sds set argument: Create a shared-disk primary server (SQL administration API)	Version 11.50.xC1
ha set idxauto argument: Replicate indexes to secondary servers (SQL administration API)	Version 11.50.xC1
ha set ipl argument: Log index builds on the primary server (SQL administration API)	Version 11.50.xC1
ha set primary argument: Define an HDR primary server (SQL administration API)	Version 11.50.xC1
ha set secondary argument: Define an HDR secondary server (SQL administration API)	Version 11.50.xC1
ha set standard argument: Convert an HDR server into a standard server (SQL administration API)	Version 11.50.xC1
ha set timeout argument: Change SD secondary server timeout (SQL administration API)	Version 11.50.xC1
onmode and d arguments: Set data-replication types (SQL administration API)	Version 11.50.xC1

Listen thread arguments

Use the following SQL administration API function arguments to control listen threads for a SOCTCP or TLITCP network protocol without interrupting existing connections.

Table 166. admin() and task() Function Arguments for Listen Thread Commands

Argument	When Added to OneDB
restart listen argument: Stop and start a listen thread dynamically (SQL administration API)	Version 11.50.xC6
start listen argument: Start a listen thread dynamically (SQL administration API)	Version 11.50.xC6
stop listen argument: Stop a listen thread dynamically (SQL administration API)	Version 11.50.xC6

Log arguments

Use the following SQL administration API function arguments to manage logical and physical logs.

Table 167. admin() and task() Function Arguments for Log Commands

Argument	When Added to OneDB
add log argument: Add a new logical log (SQL administration API) on page 747	Version 11.10.xC1
alter logmode argument: Change the database logging mode (SQL administration API) on page 751	Version 11.10.xC1
alter plog argument: Change the physical log (SQL administration API) on page 752	Version 11.10.xC1
drop log argument: Drop a logical log (SQL administration API)	Version 11.10.xC1
onmode and l arguments: Switch to the next logical log (SQL administration API)	Version 11.10.xC1

Message log arguments

Use the following SQL administration API function arguments to manage message logs.

Table 168. admin() and task() Function Arguments for Message Log Commands

Argument	When Added to OneDB
file status argument: Display the status of a message log file (SQL administration API)	Version 11.10.xC3
message log rotate argument: Rotate the message log file (SQL administration API)	Version 11.10.xC3
message log delete argument: Delete a message log file (SQL administration API)	Version 11.10.xC3
message log truncate argument: Delete the contents of a message log file (SQL administration API)	Version 11.10.xC3

Memory arguments

Use the following SQL administration API function arguments to manage memory.

Table 169. admin() and task() Function Arguments for Memory Commands

Argument	When Added to OneDB
add bufferpool argument: Add a buffer pool (SQL administration API) on page 745	Version 11.10.xC1
add memory argument: Increase shared memory (SQL administration API) on page 748	Version 11.10.xC1
onmode and a arguments: Add a shared-memory segment (SQL administration API)	Version 11.10.xC1
onmode and F arguments: Free unused memory segments (SQL administration API)	Version 11.10.xC1
onmode and n arguments: Unlock resident memory (SQL administration API)	Version 11.10.xC1

Table 169. admin() and task() Function Arguments for Memory Commands (continued)

Argument	When Added to OneDB
onmode and r arguments: Force residency of shared memory (SQL administration API)	Version 11.10.xC1
scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)	Version 11.70.xC3
scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)	Version 11.70.xC3

Mirror arguments

Use the following SQL administration API function arguments to manage mirroring.

Table 170. admin() and task() Function Arguments for Mirror Commands

Argument	When Added to OneDB
add mirror argument: Add a mirror chunk (SQL administration API) on page 749	Version 11.10.xC1
start mirroring argument: Starts storage space mirroring (SQL administration API)	Version 11.10.xC1
stop mirroring argument: Stops storage space mirroring (SQL administration API)	Version 11.10.xC1

Parallel database query (PDQ) arguments

Use the following SQL administration API function arguments to manage PDQ.

Table 171. admin() and task() Function Arguments for PDQ Commands

Argument	When Added to OneDB
onmode and D arguments: Set PDQ priority (SQL administration API)	Version 11.10.xC1
onmode and M arguments: Temporarily change decision-support memory (SQL administration API)	Version 11.10.xC1
onmode and Q arguments: Set maximum number for decision-support queries (SQL administration API)	Version 11.10.xC1
onmode and S arguments: Set maximum number of decision-support scans (SQL administration API)	Version 11.10.xC1

Server mode arguments

Use the following SQL administration API function arguments to change the server mode.

Table 172. admin() and task() Function Arguments for Server Mode Commands

Argument	When Added to OneDB
onmode and j arguments: Switch the database server to administration mode (SQL administration API)	Version 11.10.xC1
onmode and m arguments: Switch to multi-user mode (SQL administration API)	Version 11.10.xC1

Space arguments

Use the following SQL administration API function arguments to manage chunks, blobspaces, dbspaces, and sbspaces.

Table 173. admin() and task() Function Arguments for Space Commands

Argument	When Added to OneDB
add chunk argument: Add a new chunk (SQL administration API) on page 746	Version 11.10.xC1
alter chunk argument: Change chunk status to online or offline (SQL administration API) on page 750	Version 11.10.xC1
clean sbspace argument: Release unreferenced smart large objects (SQL administration API)	Version 11.10.xC1
create blobspace argument: Create a blobspace (SQL administration API)	Version 11.10.xC1
create chunk argument: Create a chunk (SQL administration API)	Version 11.10.xC1
create dbspace argument: Create a dbspace (SQL administration API)	Version 11.10.xC1
create sbspace argument: Create an sbspace (SQL administration API)	Version 11.10.xC1
create sbspace with accesstime argument: Create an sbspace that tracks access time (SQL administration API)	Version 11.70xC4
create sbspace with log argument: Create an sbspace with transaction logging (SQL administration API)	Version 11.70xC4
create tempdbspace argument: Create a temporary dbspace (SQL administration API)	Version 11.10.xC1
create tempsbspace argument: Create a temporary sbspace (SQL administration API)	Version 11.70xC4
drop blobspace argument: Drop a blobspace (SQL administration API)	Version 11.10.xC1
drop chunk argument: Drop a chunk (SQL administration API)	Version 11.10.xC1
drop dbspace argument: Drop a dbspace (SQL administration API)	Version 11.10.xC1
drop sbspace argument: Drop an sbspace (SQL administration API)	Version 11.10.xC1
drop tempdbspace argument: Drop a temporary dbspace (SQL administration API)	Version 11.10.xC1

Table 173. admin() and task() Function Arguments for Space Commands (continued)

Argument	When Added to OneDB
onmode and O arguments: Mark a disabled dbspace as down (SQL administration API)	Version 11.10.xC1
rename space argument: Rename a storage space (SQL administration API)	Version 11.10.xC1
set chunk argument: Change the status of a chunk (SQL administration API)	Version 11.10.xC1
set sbspace accesstime argument: Control access time tracking (SQL administration API)	Version 11.10.xC1
set sbspace avg_lo_size argument: Set the average size of smart large objects (SQL administration API)	Version 11.10.xC1
set sbspace logging argument: Change the logging of an sbspace (SQL administration API)	Version 11.10.xC1

Storage provisioning arguments

Also use the following SQL administration API function arguments to manage chunks, blobspaces, dbspaces, and sbspaces.

Table 174. admin() and task() Function Arguments for Storage Provisioning Space Commands

Argument	When Added to OneDB
create blobspace from storagepool argument: Create a blobspace from the storage pool (SQL administration API)	Version 11.70.xC1
create chunk from storagepool argument: Create a chunk from the storage pool (SQL administration API)	Version 11.70.xC1
create dbspace from storagepool argument: Create a dbspace from the storage pool (SQL administration API)	Version 11.70.xC1
create sbspace from storagepool argument: Create an sbspace from the storage pool (SQL administration API)	Version 11.70.xC1
create tempdbspace argument: Create a temporary dbspace (SQL administration API)	Version 11.70.xC1
create tempsbspace from storagepool argument: Create a temporary sbspace from the storage pool (SQL administration API)	Version 11.10.xC1
create tempdbspace from storagepool argument: Create a temporary dbspace from the storage pool (SQL administration API)	Version 11.70.xC1
drop blobspace to storagepool argument: Return space from an empty blobspace to the storage pool (SQL administration API)	Version 11.70.xC1

Table 174. admin() and task() Function Arguments for Storage Provisioning Space Commands (continued)

Argument	When Added to OneDB
drop chunk to storagepool argument: Return space from an empty chunk to the storage pool (SQL administration API)	Version 11.70.xC1
drop dbspace to storagepool argument: Return space from an empty dbspace to the storage pool (SQL administration API)	Version 11.70.xC1
drop sbspace to storagepool argument: Return space from an empty sbspace to the storage pool (SQL administration API)	Version 11.70.xC1
drop tempdbspace to storagepool argument: Return space from an empty temporary dbspace to the storage pool (SQL administration API)	Version 11.70.xC1
drop tempsbspace to storagepool argument: Return space from an empty temporary sbspace to the storage pool (SQL administration API)	Version 11.70.xC1
modify chunk extend argument: Extend the size of a chunk (SQL administration API)	Version 11.70.xC1
modify chunk extendable off argument: Mark a chunk as not extendable (SQL administration API)	Version 11.70.xC1
modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)	Version 11.70.xC1
modify space expand argument: Expand the size of a space (SQL administration API)	Version 11.70.xC1
modify space sp_sizes argument: Modify sizes of an extendable storage space (SQL administration API)	Version 11.70.xC1
storagepool add argument: Add a storage pool entry (SQL administration API)	Version 11.70.xC1
storagepool modify argument: Modify a storage pool entry (SQL administration API)	Version 11.70.xC1
storagepool delete argument: Delete one storage pool entry (SQL administration API)	Version 11.70.xC1
storagepool purge argument: Delete storage pool entries (SQL administration API)	Version 11.70.xC1

SQL statement cache arguments

Use the following SQL administration API function arguments to manage the SQL statement cache.

Table 175. admin() and task() Function Arguments for SQL Statement Cache Commands

Argument	When Added to OneDB
onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)	Version 11.10.xC1

Table 175. admin() and task() Function Arguments for SQL Statement Cache Commands (continued)

Argument	When Added to OneDB
onmode and W arguments: Reset statement cache attributes (SQL administration API)	Version 11.10.xC1

SQL tracing arguments

Use the following SQL administration API function arguments to manage SQL tracing.

Table 176. admin() and task() Function Arguments for SQL Tracing Commands

Argument	When Added to OneDB
set sql tracing argument: Set global SQL tracing (SQL administration API)	Version 11.10.xC1
set sql tracing database argument: Change database tracing (SQL administration API)	Version 11.50.xC3
set sql tracing session argument: Control tracing for a session (SQL administration API)	Version 11.50.xC3
set sql tracing user argument: Control tracing for users (SQL administration API)	Version 11.10.xC1
set sql user tracing argument: Set global SQL tracing for a user session (SQL administration API)	Version 11.50.xC3

Miscellaneous arguments

Use the following SQL administration API function arguments to manage miscellaneous processes.

Table 177. admin() and task() Function Arguments for Miscellaneous Commands

Argument	When Added to OneDB
onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)	Version 11.10.xC1
onmode and p arguments: Add or remove virtual processors (SQL administration API)	Version 11.10.xC1
onmode and Y arguments: Change query plan measurements for a session (SQL administration API)	Version 11.10.xC1
onmode and z arguments: Terminate a user session (SQL administration API)	Version 11.10.xC1
onmode and Z arguments: Terminate a distributed transaction (SQL administration API)	Version 11.10.xC1
print error argument: Print an error message (SQL administration API)	Version 11.10.xC1
print file info argument: Display directory or file information (SQL administration API)	Version 11.70.xC2

Table 177. admin() and task() Function Arguments for Miscellaneous Commands (continued)

Argument	When Added to OneDB
reset sysadmin argument: Move the sysadmin database (SQL administration API)	Version 11.10.xC1
scheduler argument: Stop or start the scheduler (SQL administration API)	Version 11.10.xC1

SQL administration API portal: Arguments by privilege groups

You can view a list of admin() and task() function arguments, which are sorted by privilege groups, with links to information about the arguments.

Privilege groups identify what SQL administration API commands a user can run. Some function arguments are in multiple privilege groups. Privilege groups are granted to users so that they can run the commands that they need for their jobs. By default, only user **informix** or the DBSA can run SQL administration API commands.

Use the grant admin argument to grant privileges and the revoke admin argument to revoke privileges.

- ADMIN: The user can run all SQL administration API functions.
- BAR privilege group on page 731: The user can run backup and restore functions.
- FILE privilege group on page 732: The user can manage the message log and display file information.
- GRANT privilege group on page 732: The user has the privilege to grant and revoke privileges.
- HA privilege group on page 732: The user can run high-availability functions.
- MISC privilege group on page 733: The user can administer the database server.
- MONITOR privilege group on page 738: The user can run all SQL administration API functions that only display information.
- OPERATOR: The user can run all SQL administration API functions except functions in the GRANT privilege group.
- REPLICATION privilege group on page 739: The user can run Enterprise Replication cdr utility functions.
- SQL privilege group on page 739: The user can run functions that are related to SQL statements for managing databases.
- SQLTRACE privilege group on page 739: The user can run SQL tracing functions.
- STORAGE privilege group on page 740: The user can run space-related functions.
- TENANT privilege group on page 745: The user can run tenant database functions.

BAR privilege group

The BAR privilege group includes SQL administration API function arguments to back up your databases.

Table 178. admin() and task() Function Arguments for backup and restore

Argument	Version
archive fake argument: Perform an unrecorded backup (SQL administration API) on page 752	11.50.xC1
ontape archive argument: Backup the data on your database (SQL administration API)	11.70.xC2

Table 178. admin() and task() Function Arguments for backup and restore (continued)

Argument	Version
onbar argument: Backup the storage spaces (SQL administration API)	11.70.xC2
onsmsync argument: Synchronize with the storage manager catalog (SQL administration API)	11.70.xC2

FILE privilege group

The FILE privilege group includes SQL administration API function arguments to manage message logs and display file information.

Table 179. admin() and task() Function Arguments for Message Log Commands

Argument	Version
file status argument: Display the status of a message log file (SQL administration API)	11.10.xC3
message log rotate argument: Rotate the message log file (SQL administration API)	11.10.xC3
message log delete argument: Delete a message log file (SQL administration API)	11.10.xC3
message log truncate argument: Delete the contents of a message log file (SQL administration API)	11.10.xC3
print file info argument: Display directory or file information (SQL administration API)	11.70.xC2

GRANT privilege group

The GRANT privilege group includes SQL administration API function arguments to grant or revoke privileges for running SQL administration API commands to other users.

Table 180. admin() and task() Function Arguments for granting and revoking privileges

Argument	Version
grant admin argument: Grant privileges to run SQL administration API commands	12.10.xC1
revoke admin argument: Revoke privileges to run SQL administration API commands	12.10.xC1

HA privilege group

The HA privilege group includes SQL administration API function arguments to manage high-availability clusters.

Table 181. admin() and task() Function Arguments for high-availability cluster Commands

Argument	Version
ha make primary argument: Change the mode of a secondary server (SQL administration API)	11.50.xC1

Table 181. admin() and task() Function Arguments for high-availability cluster Commands (continued)

Argument	Version
ha rss argument: Create an RS secondary server (SQL administration API)	11.50.xC1
ha rss add argument: Add an RS secondary server to a primary server (SQL administration API)	11.50.xC1
ha rss change argument: Change the password of an RS secondary server (SQL administration API)	11.50.xC1
ha rss delete argument: Delete an RS secondary server (SQL administration API)	11.50.xC1
ha sds clear argument: Stop shared-disk replication (SQL administration API)	11.50.xC1
ha sds primary argument: Convert an SD secondary server to a primary server (SQL administration API)	11.50.xC1
ha sds set argument: Create a shared-disk primary server (SQL administration API)	11.50.xC1
ha set idxauto argument: Replicate indexes to secondary servers (SQL administration API)	11.50.xC1
ha set ipl argument: Log index builds on the primary server (SQL administration API)	11.50.xC1
ha set primary argument: Define an HDR primary server (SQL administration API)	11.50.xC1
ha set secondary argument: Define an HDR secondary server (SQL administration API)	11.50.xC1
ha set standard argument: Convert an HDR server into a standard server (SQL administration API)	11.50.xC1
ha set timeout argument: Change SD secondary server timeout (SQL administration API)	11.50.xC1
onmode and d arguments: Set data-replication types (SQL administration API)	11.50.xC1

MISC privilege group

The MISC privilege group includes SQL administration function arguments to administer the database server:

- onstat on page 734
- Configuration parameters on page 734
- Data, partitions, and extents on page 735
- Listen threads on page 735
- Message log on page 735
- Memory on page 736
- PDQ on page 736
- Server mode on page 737

• SQL statement cache on page 737
• Other administrative tasks on page 737

onstat

SQL administration API function arguments to monitor the database server by running onstat commands.

Table 182. admin() and task() Function Arguments for onstat Commands

Argument	Version
onstat argument: Monitor the database server (SQL administration API)	12.10.xC1

Configuration parameters

SQL administration API function arguments to update configuration parameters.

Table 183. admin() and task() Function Arguments for Configuration Parameter Commands

Argument	Version
export config argument: Export configuration parameter values (SQL administration API)	12.10xC1
import config argument: Import configuration parameter values (SQL administration API)	12.10xC1
modify config arguments: Modify configuration parameters (SQL administration API)	12.10xC1
onmode and wf arguments: Permanently update a configuration parameter (SQL administration API)	11.10.xC1
onmode and wm arguments: Temporarily update a configuration parameter (SQL administration API)	11.10.xC1
onmode, wm, and AUTO_LRU_TUNING arguments: Change LRU tuning status (SQL administration API)	11.10.xC1
reset config argument: Revert configuration parameter value (SQL administration API)	12.10xC1
reset config all argument: Revert all dynamically updatable configuration parameter values (SQL administration API)	12.10xC1
set onconfig memory argument: Temporarily change a configuration parameter (SQL administration API)	11.50.xC3
set onconfig permanent argument: Permanently change a configuration parameter (SQL administration API)	11.50.xC3

Data, partitions, and extents

SQL administration API function arguments to manage data, partitions, and extents.

Table 184. admin() and task() Function Arguments for Data, Partition, and Extent Commands

Argument	Version
check data argument: Check data consistency (SQL administration API)	11.10.xC1
check extents argument: Check extent consistency (SQL administration API)	11.10.xC1
check partition argument: Check partition consistency (SQL administration API)	11.10.xC1
checkpoint argument: Force a checkpoint (SQL administration API)	11.10.xC1
create dbaccessdemo argument: Create the demonstration database (SQL administration API)	12.10.xC1
onmode and C arguments: Control the B-tree scanner (SQL administration API)	11.10.xC1
onmode and c arguments: Force a checkpoint (SQL administration API)	11.10.xC1
print partition argument: Print partition information (SQL administration API)	11.10.xC1
set dataskip argument: Start or stop skipping a dbspace (SQL administration API)	11.10.xC1
set index compression argument: Change index page compression (SQL administration API)	11.50.xC2

Listen threads

SQL administration API function arguments to control listen threads for a SOCTCP or TLITCP network protocol without interrupting existing connections.

Table 185. admin() and task() Function Arguments for Listen Thread Commands

Argument	Version
restart listen argument: Stop and start a listen thread dynamically (SQL administration API)	11.50.xC6
start listen argument: Start a listen thread dynamically (SQL administration API)	11.50.xC6
stop listen argument: Stop a listen thread dynamically (SQL administration API)	11.50.xC6

Message log

SQL administration API function arguments to manage message logs.

Table 186. admin() and task() Function Arguments for Message Log Commands

Argument	Version
file status argument: Display the status of a message log file (SQL administration API)	11.10.xC3
message log rotate argument: Rotate the message log file (SQL administration API)	11.10.xC3
message log delete argument: Delete a message log file (SQL administration API)	11.10.xC3
message log truncate argument: Delete the contents of a message log file (SQL administration API)	11.10.xC3

Memory

SQL administration API function arguments to manage memory.

Table 187. admin() and task() Function Arguments for Memory Commands

Argument	Version
add bufferpool argument: Add a buffer pool (SQL administration API) on page 745	11.10.xC1
add memory argument: Increase shared memory (SQL administration API) on page 748	11.10.xC1
onmode and a arguments: Add a shared-memory segment (SQL administration API)	11.10.xC1
onmode and F arguments: Free unused memory segments (SQL administration API)	11.10.xC1
onmode and n arguments: Unlock resident memory (SQL administration API)	11.10.xC1
onmode and r arguments: Force residency of shared memory (SQL administration API)	11.10.xC1
scheduler lmm enable argument: Specify automatic low memory management settings (SQL administration API)	11.70.xC3 11.50xC9
scheduler lmm disable argument: Stop automatic low memory management (SQL administration API)	11.70.xC3 11.50xC9

PDQ

SQL administration API function arguments to manage PDQ.

Table 188. admin() and task() Function Arguments for PDQ Commands

Argument	Version
onmode and D arguments: Set PDQ priority (SQL administration API)	11.10.xC1

Table 188. admin() and task() Function Arguments for PDQ Commands (continued)

Argument	Version
onmode and M arguments: Temporarily change decision-support memory (SQL administration API)	11.10.xC1
onmode and Q arguments: Set maximum number for decision-support queries (SQL administration API)	11.10.xC1
onmode and S arguments: Set maximum number of decision-support scans (SQL administration API)	11.10.xC1

Server mode

SQL administration API function arguments to change the server mode.

Table 189. admin() and task() Function Arguments for Server Mode Commands

Argument	Version
onmode and j arguments: Switch the database server to administration mode (SQL administration API)	11.10.xC1
onmode and m arguments: Switch to multi-user mode (SQL administration API)	11.10.xC1

SQL statement cache

SQL administration API function arguments to manage the SQL statement cache.

Table 190. admin() and task() Function Arguments for SQL Statement Cache Commands

Argument	Version
onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)	11.10.xC1
onmode and W arguments: Reset statement cache attributes (SQL administration API)	11.10.xC1

Other administrative tasks

SQL administration API function arguments to manage other administrative tasks.

Table 191. admin() and task() Function Arguments for other administrative task Commands

Argument	Version
alter logmode argument: Change the database logging mode (SQL administration API) on page 751	11.10.xC1

Table 191. admin() and task() Function Arguments for other administrative task Commands (continued)

Argument	Version
create dbaccessdemo argument: Create the demonstration database (SQL administration API)	12.10.xC1
onmode and e arguments: Change usage of the SQL statement cache (SQL administration API)	11.10.xC1
onmode and l arguments: Switch to the next logical log (SQL administration API)	11.10.xC1
onmode and p arguments: Add or remove virtual processors (SQL administration API)	11.10.xC1
onmode and Y arguments: Change query plan measurements for a session (SQL administration API)	11.10.xC1
onmode and z arguments: Terminate a user session (SQL administration API)	11.10.xC1
onmode and Z arguments: Terminate a distributed transaction (SQL administration API)	11.10.xC1
print error argument: Print an error message (SQL administration API)	11.10.xC1
reset sysadmin argument: Move the sysadmin database (SQL administration API)	11.10.xC1
scheduler argument: Stop or start the scheduler (SQL administration API)	11.10.xC1

MONITOR privilege group

The MONITOR privilege group includes SQL administration function arguments to monitor the message log, Enterprise Replication, and compression estimates.

Table 192. admin() and task() Function Arguments for monitoring the message log, Enterprise Replication, or compression estimates

Argument	Version
cdr error, cdr finderr, cdr list repair, cdr list replicate, cdr list replicateset, cdr list server, cdr list template, cdr stats recv, and cdr stats rqm arguments cdr argument: Administer Enterprise Replication (SQL administration API)	12.10.xC1
file status argument: Display the status of a message log file (SQL administration API)	11.10.xC3
index estimate_compression argument: Estimate index compression (SQL administration API)	12.10.xC1
print error argument: Print an error message (SQL administration API)	11.10.xC1
onstat argument: Monitor the database server (SQL administration API)	12.10.xC1

Table 192. admin() and task() Function Arguments for monitoring the message log, Enterprise Replication, or compression estimates (continued)

Argument	Version
table estimate_compression and fragment estimate_compression arguments	11.50.xC4
table or fragment arguments: Compress data and optimize storage (SQL administration API)	

REPLICATION privilege group

The REPLICATION privilege group includes SQL administration API function arguments to manage Enterprise Replication.

Table 193. admin() and task() Function Arguments for Enterprise Replication Commands

Argument	Version
cdr argument: Administer Enterprise Replication (SQL administration API)	11.50.xC3

SQL privilege group

The SQL privilege group includes SQL administration API function arguments to create and drop databases and view error messages.

Table 194. admin() and task() Function arguments for databases and error messages

Argument	Version
create database argument: Create a database (SQL administration API)	11.70.xC2
create dbaccessdemo argument: Create the demonstration database (SQL administration API)	12.10.xC1
drop database argument: Drop a database (SQL administration API)	11.70.xC2
print error argument: Print an error message (SQL administration API)	11.10.xC1

SQLTRACE privilege group

The SQLTRACE privilege group includes SQL administration API function arguments to manage SQL tracing.

Table 195. admin() and task() Function Arguments for SQL Tracing Commands

Argument	Version
set sql tracing argument: Set global SQL tracing (SQL administration API)	11.10.xC1
set sql tracing database argument: Change database tracing (SQL administration API)	11.50.xC3

Table 195. admin() and task() Function Arguments for SQL Tracing Commands (continued)

Argument	Version
set sql tracing session argument: Control tracing for a session (SQL administration API)	11.50.xC3
set sql tracing user argument: Control tracing for users (SQL administration API)	11.10.xC1
set sql user tracing argument: Set global SQL tracing for a user session (SQL administration API)	11.50.xC3

STORAGE privilege group

The STORAGE privilege group includes SQL administration API function arguments for managing the following aspects of storage:

- Storage space encryption argument on page 740
- Automatic table storage location arguments on page 740
- Compression on page 741
- Logical and physical logs on page 741
- Mirroring on page 742
- Storage spaces on page 742
- Storage provisioning on page 743

Storage space encryption argument

SQL administration API function argument to change the master encryption key for storage space encryption.

Table 196. admin() and task() Function Arguments for storage space encryption command

Argument	Version
master_key reset argument: Change the keystore password (SQL administration API)	12.10.xC8

Automatic table storage location arguments

SQL administration API function arguments to manage the list of dbspaces that store automatically allocated fragments.

Table 197. admin() and task() Function Arguments for table storage commands

Argument	Version
autolocate database add argument: Add a dbspace to the dbspace list (SQL administration API) on page 753	12.10.xC3
autolocate database anywhere argument: Add all dbspaces to the dbspace list (SQL administration API) on page 754	12.10.xC3

Table 197. admin() and task() Function Arguments for table storage commands (continued)

Argument	Version
autolocate database argument: Specify dbspaces for automatic location and fragmentation (SQL administration API) on page 754	12.10.xC3
autolocate database off argument: Disable automatic fragmentation for a database (SQL administration API) on page 755	12.10.xC3
autolocate database remove argument: Remove a dbspace from the dbspace list (SQL administration API) on page 755	12.10.xC3

Compression

SQL administration API function arguments to manage the compression of data and to optimize storage.

Table 198. admin() and task() Function Arguments for Compression Commands

Argument	Version
index compress repack shrink arguments: Optimize the storage of B-tree indexes (SQL administration API)	12.10.xC1
index estimate_compression argument: Estimate index compression (SQL administration API)	12.10.xC1
table or fragment arguments: Compress data and optimize storage (SQL administration API)	11.50.xC4
purge compression dictionary arguments: Remove compression dictionaries (SQL administration API)	11.50.xC4

For an overview of compression and storage optimization commands, see Table and fragment compress and uncompress operations (SQL administration API).

Logical and physical logs

SQL administration API function arguments to manage logical and physical logs.

Table 199. admin() and task() Function Arguments for Log Commands

Argument	Version
add log argument: Add a new logical log (SQL administration API) on page 747	11.10.xC1
alter logmode argument: Change the database logging mode (SQL administration API) on page 751	11.10.xC1
alter plog argument: Change the physical log (SQL administration API) on page 752	11.10.xC1

Table 199. admin() and task() Function Arguments for Log Commands (continued)

Argument	Version
drop log argument: Drop a logical log (SQL administration API)	11.10.xC1

Mirroring

SQL administration API function arguments to manage mirroring.

Table 200. admin() and task() Function Arguments for Mirror Commands

Argument	Version
add mirror argument: Add a mirror chunk (SQL administration API) on page 749	11.10.xC1
start mirroring argument: Starts storage space mirroring (SQL administration API)	11.10.xC1
stop mirroring argument: Stops storage space mirroring (SQL administration API)	11.10.xC1

Storage spaces

SQL administration API function arguments to manage chunks, blobspaces, dbspaces, and sbspaces.

Table 201. admin() and task() Function Arguments for Space Commands

Argument	Version
add chunk argument: Add a new chunk (SQL administration API) on page 746	11.10.xC1
alter chunk argument: Change chunk status to online or offline (SQL administration API) on page 750	11.10.xC1
clean sbspace argument: Release unreferenced smart large objects (SQL administration API)	11.10.xC1
create blobspace argument: Create a blobspace (SQL administration API)	11.10.xC1
create chunk argument: Create a chunk (SQL administration API)	11.10.xC1
create dbaccessdemo argument: Create the demonstration database (SQL administration API)	12.10.xC1
create dbspace argument: Create a dbspace (SQL administration API)	11.10.xC1
create sbspace argument: Create an sbspace (SQL administration API)	11.10.xC1
create sbspace with accesstime argument: Create an sbspace that tracks access time (SQL administration API)	11.70.xC4
create sbspace with log argument: Create an sbspace with transaction logging (SQL administration API)	11.70.xC4

Table 201. admin() and task() Function Arguments for Space Commands (continued)

Argument	Version
create tempdbspace argument: Create a temporary dbspace (SQL administration API)	11.10.xC1
create tempsbspace argument: Create a temporary sbspace (SQL administration API)	11.70.xC4
drop blobspace argument: Drop a blobspace (SQL administration API)	11.10.xC1
drop chunk argument: Drop a chunk (SQL administration API)	11.10.xC1
drop dbspace argument: Drop a dbspace (SQL administration API)	11.10.xC1
drop sbspace argument: Drop an sbspace (SQL administration API)	11.10.xC1
drop tempdbspace argument: Drop a temporary dbspace (SQL administration API)	11.10.xC1
onmode and O arguments: Mark a disabled dbspace as down (SQL administration API)	11.10.xC1
print error argument: Print an error message (SQL administration API)	11.10.xC1
rename space argument: Rename a storage space (SQL administration API)	11.10.xC1
set chunk argument: Change the status of a chunk (SQL administration API)	11.10.xC1
set sbspace accesstime argument: Control access time tracking (SQL administration API)	11.10.xC1
set sbspace avg_lo_size argument: Set the average size of smart large objects (SQL administration API)	11.10.xC1
set sbspace logging argument: Change the logging of an sbspace (SQL administration API)	11.10.xC1

Storage provisioning

SQL administration API function arguments to manage chunks, blobspaces, dbspaces, and sbspaces from storage pools.

Table 202. admin() and task() Function Arguments for Storage Provisioning Space Commands

Argument	Version
create blobspace from storagepool argument: Create a blobspace from the storage pool (SQL administration API)	11.70.xC1
create chunk from storagepool argument: Create a chunk from the storage pool (SQL administration API)	11.70.xC1
create dbspace from storagepool argument: Create a dbspace from the storage pool (SQL administration API)	11.70.xC1

Table 202. admin() and task() Function Arguments for Storage Provisioning Space Commands (continued)

Argument	Version
create plogspace: Create a plogspace (SQL administration API)	12.10.xC3
create sbspace from storagepool argument: Create an sbspace from the storage pool (SQL administration API)	11.70.xC1
create tempdbspace argument: Create a temporary dbspace (SQL administration API)	11.70.xC1
create tempsbspace from storagepool argument: Create a temporary sbspace from the storage pool (SQL administration API)	11.10.xC1
create tempdbspace from storagepool argument: Create a temporary dbspace from the storage pool (SQL administration API)	11.70.xC1
drop blobspace to storagepool argument: Return space from an empty blobspace to the storage pool (SQL administration API)	11.70.xC1
drop chunk to storagepool argument: Return space from an empty chunk to the storage pool (SQL administration API)	11.70.xC1
drop dbspace to storagepool argument: Return space from an empty dbspace to the storage pool (SQL administration API)	11.70.xC1
drop plogspace: Drop the plogspace (SQL administration API)	12.10.xC3
drop sbspace to storagepool argument: Return space from an empty sbspace to the storage pool (SQL administration API)	11.70.xC1
drop tempdbspace to storagepool argument: Return space from an empty temporary dbspace to the storage pool (SQL administration API)	11.70.xC1
drop tempsbspace to storagepool argument: Return space from an empty temporary sbspace to the storage pool (SQL administration API)	11.70.xC1
modify chunk extend argument: Extend the size of a chunk (SQL administration API)	11.70.xC1
modify chunk extendable off argument: Mark a chunk as not extendable (SQL administration API)	11.70.xC1
modify chunk extendable argument: Mark a chunk as extendable (SQL administration API)	11.70.xC1
modify space expand argument: Expand the size of a space (SQL administration API)	11.70.xC1
modify space sp_sizes argument: Modify sizes of an extendable storage space (SQL administration API)	11.70.xC1
storagepool add argument: Add a storage pool entry (SQL administration API)	11.70.xC1

Table 202. admin() and task() Function Arguments for Storage Provisioning Space Commands (continued)

Argument	Version
storagepool modify argument: Modify a storage pool entry (SQL administration API)	11.70.xC1
storagepool delete argument: Delete one storage pool entry (SQL administration API)	11.70.xC1
storagepool purge argument: Delete storage pool entries (SQL administration API)	11.70.xC1

TENANT privilege group

The TENANT privilege group includes SQL administration API function arguments to manage tenant databases.

Table 203. admin() and task() Function Arguments for Tenant Database Commands

Argument	Version
tenant create argument: Create a tenant database (SQL Administration API)	12.10.xC4
tenant drop argument: Drop a tenant database (SQL Administration API)	12.10.xC4
tenant update argument: Modify tenant database properties (SQL Administration API)	12.10.xC4

add bufferpool argument: Add a buffer pool (SQL administration API)

Use the **add bufferpool** argument with the admin() or task() function to create a buffer pool.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "add bufferpool", "page_size" );
EXECUTE FUNCTION { admin | task } ( "add bufferpool", "page_size", "buffers" [ , "lrus" [ , "max_dirty" [ , "min_dirty" ] ] ] );
```

Element	Description	Key Considerations
buffers	The number of buffers. The default is 10000 buffers.	Each buffer is the size of the operating system page.
lrus	The number of LRU queues. The default is 8.	The maximum for 32-bit platforms is 128, and for 64-bit platforms is 512.
max_dirty	The percentage of modified pages in the LRU queues at which the queue is cleaned. If a field is specified out of the range of values, the default of 60.00 percent is set.	

Element	Description	Key Considerations
min_dirty	The percentage of modified pages in the LRU queues at which page cleaning is not mandatory. Page cleaners might continue cleaning beyond this point under some circumstances. If a field is specified out of the range of values, the default of 80.00 percent is set.	
page_size	The page size in KB.	The page size must be an integral multiple of the default page size, and cannot be greater than 16 KB. On Windows™, the page size is always 4 KB.

Usage

Use **add bufferpool** argument to create a buffer pool for a page size that does not already have a buffer pool. All other characteristics of the buffer pool that you create are set to the values of the fields in the default line of the BUFFERPOOL configuration parameter.

This function is equivalent to the onparams -b -g command and the BUFFERPOOL configuration parameter.

Example

Example

The following example adds a buffer pool with a page size of 8 KB:

```
EXECUTE FUNCTION task("add bufferpool","8");
```

Example

Example

The following example adds a buffer pool with a page size of 2 KB, 50,000 buffers, 8 LRU queues, an LRU maximum of 60 percent, and an LRU minimum of 50 percent:

```
EXECUTE FUNCTION task("add bufferpool","2","50000","8","60.0","50.0");
```

add chunk argument: Add a new chunk (SQL administration API)

Use the **add chunk** argument with the admin() or task() function to add a chunk to a dbspace or blobspace.

Syntax

EXECUTE FUNCTION { **admin** | **task** } (**"add chunk"**, *"space_name"*, *"path_name"* [, *"disk_size"* [, *"offset"* [, *"mirror_path"* [, *"mirror_offset"*]]]]);

Element	Description	Key Considerations
disk_size	The amount of disk space to add in kilobytes.	See admin() and task() Argument Size Specifications on page 718.
mirror_offset	The location of the mirror chunk.	
mirror_path	The path to the mirror chunk.	If you are adding a chunk to a mirrored storage space, you must also add a mirror chunk.
offset	The location of the new chunk.	
path_name	The path of the added disk space.	
space_name	The name of the dbspace, blobspace, or sbspace to which you are adding disk space.	

Usage

The size of the chunk must be equal to or greater than 1000 KB and a multiple of the page size. The starting offset plus the chunk size cannot exceed the maximum chunk size. The maximum offset is 4 TB.

This function is equivalent to the **onspaces -a** command.

Example

Example

The following example adds a 5 MB chunk of raw disk space, at an offset of 5200 kilobytes, to a dbspace named **dbspc3**:

```
EXECUTE FUNCTION task("add chunk", "dbspc3","\\.\e:","5120","5200");
```

The following example adds a 10 MB mirror chunk to a blobspace named **blobsp3** with an offset of 200 kilobytes for both the primary and mirror chunks:

```
EXECUTE FUNCTION task("add chunk","blobsp3","/dev/raw_dev1","10240",
"200","/dev/raw_dev2","200");
```

add log argument: Add a new logical log (SQL administration API)

Use the **add log** argument with the admin() or task() function to add a logical log to a dbspace.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "add log", "dbspace" [ , "size" [ , { 1 | "count" } [ , after_current_flag ] ] ] );
```

Element	Description	Key Considerations
after_current_flag	Whether to add the new log after the current log or after the last logical log (default).	Possible values are: • `1` = Add the new log after the current log. • `0` = Add the new log after the last log.
count	The number of log files to create. The default is 1.	The number must not cause the total number of logical-log files to exceed 32,767.
dbspace	The name of the dbspace in which to insert a logical-log file.	You can add a log file to a dbspace only if the database server has adequate contiguous space. You can add a log file during a backup. You cannot add a log file to a blobspace or sbspace.
size	The size in kilobytes of the new logical-log file. The default is the size specified by the LOGSIZE configuration parameter.	This value must be an unsigned integer greater than or equal to 200 KB. Also see admin() and task() Argument Size Specifications on page 718.

Usage

The newly added log files have a status of **A** and are immediately available for use. Use onstat -l to view the status of your logical-log files. It is recommended that you take a level-0 backup of the root dbspace and the dbspace that contains the log file as soon as possible after running this function.

By default, the new log file is added after the last logical log. Include `1` as the fifth argument to insert the logical-log file after the current log file.

This function resembles the **onparams -a -d** command, which can add a single logical-log file. You can add multiple logical-log files to the specified dbspace, however, with a single invocation of this function.

Example

Example

The command in the following example adds three logical logs after the current log, each with a size of 5 MB:

```
EXECUTE FUNCTION task ("add log","logdbs","5M",3,1);
```

add memory argument: Increase shared memory (SQL administration API)

Use the **add memory** argument with the admin() or task() function to add to the virtual portion of shared memory.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "add memory", "memory_size" );
```

Element	Description	Key Considerations
memory_size	The size, in kilobytes, of the new virtual shared-memory segment.	This value must not exceed the operating system limit for the size of shared-memory segments. Also see admin() and task() Argument Size Specifications on page 718.

Usage

This size defaults to the SHMADD configuration parameter.

This function is equivalent to the **onmode -a** command.

Example

Example

The following example adds 500 KB of virtual shared-memory:

```
EXECUTE FUNCTION task("add memory","500");
```

add mirror argument: Add a mirror chunk (SQL administration API)

Use the **add mirror** argument with the admin() or task() function to add a mirror chunk to a dbspace.

Syntax

```
>>-EXECUTE FUNCTION--+-admin-+--(------------------------------>
                     '-task--'

>--"add mirror"--,--"space_name"--,--"path_name"--,------------->

>--"offset"--,--"mirror_path"--,--"mirror_offset"--)--;--------><
```

Element	Description	Key Considerations
mirror_path	The disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that performs the mirroring.	
mirror_offset	The offset to reach the mirrored chunk of the newly mirrored dbspace, blobspace, or sbspace.	See admin() and task() Argument Size Specifications on page 718.

Element	Description	Key Considerations
offset	The offset into the disk partition or into the unbuffered device in kilobytes to reach the initial chunk of the newly mirrored dbspace, blobspace, or sbspace.	See admin() and task() Argument Size Specifications on page 718.
path_name	The disk partition or unbuffered device of the initial chunk of the dbspace, blobspace, or sbspace that you want to mirror.	
space_name	The name of a dbspace, blobspace, or sbspace to mirror.	

Usage

This function is equivalent to the **onspaces -m** command.

Example

Example

The following example adds a mirror chunk to a blobspace named **blobsp3**:

```
EXECUTE FUNCTION task("add mirror","blobsp3","/dev/raw_dev1",
"10240","/dev/raw_dev2","200");
```

alter chunk argument: Change chunk status to online or offline (SQL administration API)

Use the **alter chunk** argument with the admin() or task() function to bring a chunk online or take a chunk offline in a dbspace, blobspace, or sbspace.

Syntax

```
EXECUTE FUNCTION { admin | task } ( { "alter chunk offline" | "alter chunk online" } , "space_name", "path_ name", "offset" );
```

Element	Description	Key Considerations
space_name	The name of the blobspace, dbspace, or sbspace.	
path_name	The disk partition or unbuffered device of the chunk.	
offset	The offset (in kilobytes) into the disk partition or unbuffered device to reach the chunk. The default is `0`.	See admin() and task() Argument Size Specifications on page 718.

Usage

The chunk must be in a mirrored pair, or a non-primary chunk within a noncritical dbspace.

Use the **alter chunk online** argument to change the chunk status to online.

Use the **alter chunk offline** argument to change the chunk status to offline.

This function is equivalent to the **onspaces -s** command.

Example

Example

The following example brings a chunk in a space named `dbspace4` online:

```
EXECUTE FUNCTION task("alter chunk online","dbspace4","/dev/raw_dev1","0");
```

alter logmode argument: Change the database logging mode (SQL administration API)

Use the **alter logmode** argument with the admin() or task() function to change the database logging mode to ANSI, buffered, non-logging, or unbuffered.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "alter logmode", "database_name", { "a" | "b" | "n" | "u" } );
```

Element	Description	Key Considerations
database_name	The name of the database with the logging mode that you want to alter.	

Usage

Unlike when you change the database logging mode with the **ondblog** utilitiy, when you use this function, the database remains accessible, and a level-0 backup is not always required. Ensure that no other session is active before running this function or it will fail.

Use the **"a"** argument to change the database logging to be ANSI compliant. After you create or convert a database to ANSI mode, you cannot change it back to any of the other logging modes.

Use the **"b"** argument to change the database logging to be buffered, so that transaction information is written to a buffer before it is written to a logical log.

Use the **"n"** argument to change the database logging to be non-logging, so that no database transactions are logged. You must perform a level-0 backup prior to using this argument.

Use the **"u"** argument to change the database logging to be unbuffered, so that data is not written to a buffer before it is written to a logical log.

Example

Example

The following example changes the logging mode of a database named **employee** to unbuffered logging:

```
EXECUTE FUNCTION task("alter logmode","employee","u");
```

alter plog argument: Change the physical log (SQL administration API)

Use the **alter plog** argument with the admin() or task() function to change the location and size of the physical log.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "alter plog", "dbspace" [ , "phys_log_size" ] );
```

Element	Description	Key Considerations
dbspace	The location of the physical log.	The space allocated for the physical log must be contiguous.
phys_log_size	The size, specified in kilobytes, of the physical log.	See admin() and task() Argument Size Specifications on page 718.

Usage

To change only the size, specify the current dbspace of the physical log.

This function is equivalent to the **onparams -p** command.

Example

Example

The following example moves the physical log to a dbspace called **phsdbs**:

```
EXECUTE FUNCTION task ("alter plog","physdbs","49 M");
```

archive fake argument: Perform an unrecorded backup (SQL administration API)

Use the **archive fake** argument with the admin() or task() function to perform a backup operation to clone the data in a server without creating a persistent backup that could be used to perform a restore.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "archive fake" );
```

Usage

Use this function to populate the secondary server in a High-Availability Data Replication pair.

Example

Example

The following example starts an unrecorded backup:

```
EXECUTE FUNCTION task("archive fake");
```

autolocate database add argument: Add a dbspace to the dbspace list (SQL administration API)

Use the **autolocate database add** argument with the admin() or task() function to add a dbspace to the list of available dbspaces for the automatic location and fragmentation of tables for the specified database.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "autolocate database add", "database_name" , "dbspace" );
```

Element	Description	Key Considerations
database_name	Name of the database	
dbspace	Name of a dbspace to add to the list of names of the dbspaces in which the database server can automatically create fragments.	The dbspace must exist.

Usage

The AUTOLOCATE configuration parameter or session environment variable must be set to a positive integer.

The list of available dbspaces is stored in the **sysautolocate** system catalog table.

Example

Example

The following command adds the dbspace **dbspace9** to the list of available dbspaces for automatic location and fragmentation for tables in the **customer** database.

```
EXECUTE FUNCTION task("autolocate database add", "customer", "dbspace9");
```

autolocate database anywhere argument: Add all dbspaces to the dbspace list (SQL administration API)

Use the **autolocate database anywhere** argument with the admin() or task() function to specify that the database server can use any non-critical dbspace for the automatic location and fragmentation of tables for the specified database.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "autolocate database anywhere", "database_name" );
```

Element	Description	Key Considerations
database_name	Name of the database	Cannot be the name of a tenant database.

Usage

This command replaces any previous list of dbspaces with a list of all available dbspaces. Dbspaces that are dedicated to tenant database are not available. The list of available dbspaces is stored in the **sysautolocate** system catalog table.

The AUTOLOCATE configuration parameter or session environment variable must be set to a positive integer.

Example

Example

The following command adds all non-critical dbspaces to the list of available dbspaces for automatic location and fragmentation for tables in the **potential_cust** database:

```
EXECUTE FUNCTION task("autolocate database anywhere", "potential_cust");
```

autolocate database argument: Specify dbspaces for automatic location and fragmentation (SQL administration API)

Use the **autolocate database** argument with the admin() or task() function to specify the list of available dbspaces for the automatic location and fragmentation of tables for the specified database.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "autolocate database", "database_name" , "dbspace_list" );
```

Element	Description	Key Considerations
database_name	Name of the database	Cannot be the name of a tenant database.
dbspace_list	A comma-separated list of names of the dbspaces in which the database server can automatically create fragments.	The dbspaces must exist. The dbspaces cannot be dedicated to a tenant database.

Usage

The AUTOLOCATE configuration parameter or session environment variable must be set to a positive integer.

By default, all dbspaces are available. The list of available dbspaces is stored in the **sysautolocate** system catalog table.

Example

Example

The following command limits the list of available dbspaces for automatic location and fragmentation for tables in the **customer** database:

```
EXECUTE FUNCTION task("autolocate database", "customer",
 "dbspace1,dbspace2,dbspace4,dbspace8");
```

autolocate database off argument: Disable automatic fragmentation for a database (SQL administration API)

Use the **autolocate database off** argument with the admin() or task() function to disable the automatic location and fragmentation of tables for a specified database.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "autolocate database off", "database_name" );
```

Element	Description	Key Considerations
database_name	Name of the database	

Usage

New tables that you create in the specified database are stored in the same dbspace as the database and are not fragmented. Existing tables that were automatically fragmented are not allocated new fragments as the table grows.

Example

Example

The following command disables automatic location and fragmentation of tables in the **customer_old** database:

```
EXECUTE FUNCTION task("autolocate database off", "customer_old");
```

autolocate database remove argument: Remove a dbspace from the dbspace list (SQL administration API)

Use the **autolocate database remove** argument with the admin() or task() function to remove a dbspace from the list of available dbspaces into which the database server can automatically locate and fragment tables for the specified database.

Syntax

```
EXECUTE FUNCTION { admin | task } ( "autolocate database remove", "database_name" , "dbspace" );
```

Element	Description	Key Considerations
database_ name	Name of the database	
dbspace	Name of the dbspace to remove from the list of names of dbspaces in which the database server can automatically create fragments.	The dbspace must exist.

Usage

The AUTOLOCATE configuration parameter or session environment variable must be set to a positive integer.

The list of available dbspaces is stored in the **sysautolocate** system catalog table.

Example

Example

The following command removes **dbspace1** from the list of available dbspaces for the **customer** database.

```
EXECUTE FUNCTION task("autolocate database remove", "customer", "dbspace1");
```

Index

Special Characters

Numerics

A

B

C

E

F

G

N

O

S

T

U