data together with the UNDO log, and then applies the REDO log to play back all changes up to the restoration point.

`RedoBuffer` sets the size of the buffer in which the REDO log is written. The default value is 32MB; the minimum value is 1MB.

If this buffer is too small, the `NDB` storage engine issues error code 1221 (`REDO log buffers overloaded`). For this reason, you should exercise care if you attempt to decrease the value of `RedoBuffer` as part of an online change in the cluster's configuration.

`ndbmtd` allocates a separate buffer for each LDM thread (see `ThreadConfig`). For example, with 4 LDM threads, an `ndbmtd` data node actually has 4 buffers and allocates `RedoBuffer` bytes to each one, for a total of `4 * RedoBuffer` bytes.

- `EventLogBufferSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	8192
Range	0 - 64K
Restart Type	**System Restart:** Requires a complete shutdown and restart of the cluster. (NDB 8.0.13)

Controls the size of the circular buffer used for NDB log events within data nodes.

Controlling log messages. In managing the cluster, it is very important to be able to control the number of log messages sent for various event types to `stdout`. For each event category, there are 16 possible event levels (numbered 0 through 15). Setting event reporting for a given event category to level 15 means all event reports in that category are sent to `stdout`; setting it to 0 means that no event reports in that category are made.

By default, only the startup message is sent to `stdout`, with the remaining event reporting level defaults being set to 0. The reason for this is that these messages are also sent to the management server's cluster log.

An analogous set of levels can be set for the management client to determine which event levels to record in the cluster log.

- `LogLevelStartup`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	1
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The reporting level for events generated during startup of the process.

The default level is 1.

- `LogLevelShutdown`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The reporting level for events generated as part of graceful shutdown of a node.

The default level is 0.

- `LogLevelStatistic`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The reporting level for statistical events such as number of primary key reads, number of updates, number of inserts, information relating to buffer usage, and so on.

The default level is 0.

- `LogLevelCheckpoint`

Version (or later)	NDB 8.0.13
Type or units	log level
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

The reporting level for events generated by local and global checkpoints.

The default level is 0.

- `LogLevelNodeRestart`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The reporting level for events generated during node restart.

The default level is 0.

- `LogLevelConnection`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The reporting level for events generated by connections between cluster nodes.

The default level is 0.

- `LogLevelError`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

The reporting level for events generated by errors and warnings by the cluster as a whole. These errors do not cause any node failure but are still considered worth reporting.

The default level is 0.

- `LogLevelCongestion`

Version (or later)	NDB 8.0.13
Type or units	level
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The reporting level for events generated by congestion. These errors do not cause node failure but are still considered worth reporting.

The default level is 0.

- `LogLevelInfo`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 15
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The reporting level for events generated for information about the general state of the cluster.

The default level is 0.

- `MemReportFrequency`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a
	rolling restart

	of the cluster. (NDB 8.0.13)

This parameter controls how often data node memory usage reports are recorded in the cluster log; it is an integer value representing the number of seconds between reports.

Each data node's data memory and index memory usage is logged as both a percentage and a number of 32 KB pages of `DataMemory`, as set in the `config.ini` file. For example, if `DataMemory` is equal to 100 MB, and a given data node is using 50 MB for data memory storage, the corresponding line in the cluster log might look like this:

```
2006-12-24 01:18:16 [MgmSrvr] INFO -- Node 2: Data usage is 50%(1280 32K pages of total 2560)
```

`MemReportFrequency` is not a required parameter. If used, it can be set for all cluster data nodes in the `[ndbd default]` section of `config.ini`, and can also be set or overridden for individual data nodes in the corresponding `[ndbd]` sections of the configuration file. The minimum value—which is also the default value—is 0, in which case memory reports are logged only when memory usage reaches certain percentages (80%, 90%, and 100%), as mentioned in the discussion of statistics events in Section 23.6.3.2, “NDB Cluster Log Events”.

- `StartupStatusReportFrequency`

Version (or later)	NDB 8.0.13
Type or units	seconds
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

When a data node is started with the `--initial`, it initializes the redo log file during Start Phase 4 (see Section 23.6.4, “Summary of NDB Cluster Start Phases”). When very large values are set for `NoOfFragmentLogFiles`, `FragmentLogFileSize`, or both, this initialization can take a long time. You can force reports on the progress of this process to be logged periodically, by means of the `StartupStatusReportFrequency` configuration parameter. In this case, progress is reported in the cluster log, in terms of both the number of files and the amount of space that have been initialized, as shown here:

```
2009-06-20 16:39:23 [MgmSrvr] INFO -- Node 1: Local redo log file initialization status:
#Total files: 80, Completed: 60
#Total MBytes: 20480, Completed: 15557
2009-06-20 16:39:23 [MgmSrvr] INFO -- Node 2: Local redo log file initialization status:
#Total files: 80, Completed: 60
#Total MBytes: 20480, Completed: 15570
```

These reports are logged each `StartupStatusReportFrequency` seconds during Start Phase 4. If `StartupStatusReportFrequency` is 0 (the default), then reports are written to the cluster log only when at the beginning and at the completion of the redo log file initialization process.

Data Node Debugging Parameters

The following parameters are intended for use during testing or debugging of data nodes, and not for use in production.

- `DictTrace`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	undefined
Range	0 - 100
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

It is possible to cause logging of traces for events generated by creating and dropping tables using `DictTrace`. This parameter is useful only in debugging NDB kernel code. `DictTrace` takes an integer value. 0 is the default, and means no logging is performed; 1 enables trace logging, and 2 enables logging of additional `DBDICT` debugging output.

- `WatchdogImmediateKill`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

You can cause threads to be killed immediately whenever watchdog issues occur by enabling the `WatchdogImmediateKill` data node configuration parameter. This parameter should be used only when debugging or troubleshooting, to obtain trace files reporting exactly what was occurring the instant that execution ceased.

Backup parameters. The `[ndbd]` parameters discussed in this section define memory buffers set aside for execution of online backups.

- `BackupDataBufferSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	16M
Range	512K - 4294967039 (0xFFFFFEFF)
Deprecated	Yes (in NDB 7.6)
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

In creating a backup, there are two buffers used for sending data to the disk. The backup data buffer is used to fill in data recorded by scanning a node's tables. Once this buffer has been filled to the level specified as `BackupWriteSize`, the pages are sent to disk. While flushing data to disk, the backup process can continue filling this buffer until it runs out of space. When this happens, the backup process pauses the scan and waits until some disk writes have completed freeing up memory so that scanning may continue.

The default value for this parameter is 16MB. The minimum is 512K.

- `BackupDiskWriteSpeedPct`

Version (or later)	NDB 8.0.13
Type or units	percent
Default	50
Range	0 - 90
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

During normal operation, data nodes attempt to maximize the disk write speed used for local checkpoints and backups while remaining within the bounds set by `MinDiskWriteSpeed` and `MaxDiskWriteSpeed`. Disk write throttling gives each LDM thread an equal share of the total budget. This allows parallel LCPs to take place without exceeding the disk I/O budget. Because a backup is executed by only one LDM thread, this effectively caused a budget cut, resulting in longer backup completion times, and—if the rate of change is sufficiently high—in failure to complete the backup when the backup log buffer fill rate is higher than the achievable write rate.

This problem can be addressed by using the `BackupDiskWriteSpeedPct` configuration parameter, which takes a value in the range 0-90 (inclusive) which is interpreted as the percentage of the node's maximum write rate budget that is reserved prior to sharing out the remainder of the budget among LDM threads for LCPs. The LDM thread running the backup receives the whole write rate budget for the backup, plus its (reduced) share of the write rate budget for local checkpoints.

The default value for this parameter is 50 (interpreted as 50%).

- `BackupLogBufferSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	16M
Range	2M - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

The backup log buffer fulfills a role similar to that played by the backup data buffer, except that it is used for generating a log of all table writes made during execution of the backup. The same principles apply for writing these pages as with the backup data buffer, except that when there is no more space in the backup log buffer, the backup fails. For that reason, the size of the backup log buffer must be large enough to handle the load caused by write activities while the backup is being made. See Section 23.6.8.3, “Configuration for NDB Cluster Backups”.

The default value for this parameter should be sufficient for most applications. In fact, it is more likely for a backup failure to be caused by insufficient disk write speed than it is for the backup log buffer to become full. If the disk subsystem is not configured for the write load caused by applications, the cluster is unlikely to be able to perform the desired operations.

It is preferable to configure cluster nodes in such a manner that the processor becomes the bottleneck rather than the disks or the network connections.

The default value for this parameter is 16MB.

- `BackupMemory`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	32M
Range	0 - 4294967039 (0xFFFFFEFF)
Deprecated	Yes (in NDB 7.4)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter is deprecated, and subject to removal in a future version of NDB Cluster. Any setting made for it is ignored.

- `BackupReportFrequency`

Version (or later)	NDB 8.0.13
Type or units	seconds
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter controls how often backup status reports are issued in the management client during a backup, as well as how often such reports are written to the cluster log (provided cluster event

logging is configured to permit it—see Logging and checkpointing). `BackupReportFrequency` represents the time in seconds between backup status reports.

The default value is 0.

- `BackupWriteSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	256K
Range	32K - 4294967039 (0xFFFFFEFF)
Deprecated	Yes (in NDB 7.6)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter specifies the default size of messages written to disk by the backup log and backup data buffers.

The default value for this parameter is 256KB.

- `BackupMaxWriteSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	1M
Range	256K - 4294967039 (0xFFFFFEFF)
Deprecated	Yes (in NDB 7.6)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter specifies the maximum size of messages written to disk by the backup log and backup data buffers.

The default value for this parameter is 1MB.

- `CompressedBackup`

Version (or later)	NDB 8.0.13
Type or units	boolean

Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Enabling this parameter causes backup files to be compressed. The compression used is equivalent to `gzip --fast`, and can save 50% or more of the space required on the data node to store uncompressed backup files. Compressed backups can be enabled for individual data nodes, or for all data nodes (by setting this parameter in the `[ndbd default]` section of the `config.ini` file).

Important

You cannot restore a compressed backup to a cluster running a MySQL version that does not support this feature.

The default value is `0` (disabled).

- `RequireEncryptedBackup`

Version (or later)	NDB 8.0.22
Type or units	integer
Default	0
Range	0 - 1
Added	NDB 8.0.22
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

If set to 1, backups must be encrypted. While it is possible to set this parameter for each data node individually, it is recommended that you set it in the `[ndbd default]` section of the `config.ini` global configuration file. For more information about performing encrypted backups, see Section 23.6.8.2, “Using The NDB Cluster Management Client to Create a Backup”.

Added in NDB 8.0.22.

Note

The location of the backup files is determined by the `BackupDataDir` data node configuration parameter.

Additional requirements. When specifying these parameters, the following relationships must hold true. Otherwise, the data node cannot start.

- `BackupDataBufferSize >= BackupWriteSize + 188KB`
- `BackupLogBufferSize >= BackupWriteSize + 16KB`
- `BackupMaxWriteSize >= BackupWriteSize`

NDB Cluster Realtime Performance Parameters

The `[ndbd]` parameters discussed in this section are used in scheduling and locking of threads to specific CPUs on multiprocessor data node hosts.

Note

To make use of these parameters, the data node process must be run as system root.

- `BuildIndexThreads`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	128
Range	0 - 128
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter determines the number of threads to create when rebuilding ordered indexes during a system or node start, as well as when running `ndb_restore --rebuild-indexes`. It is supported only when there is more than one fragment for the table per data node (for example, when `COMMENT="NDB_TABLE=PARTITION_BALANCE=FOR_RA_BY_LDM_X_2"` is used with `CREATE TABLE`).

Setting this parameter to 0 (the default) disables multithreaded building of ordered indexes.

This parameter is supported when using `ndbd` or `ndbmtd`.

You can enable multithreaded builds during data node initial restarts by setting the `TwoPassInitialNodeRestartCopy` data node configuration parameter to `TRUE`.

- `LockExecuteThreadToCPU`

Version (or later)	NDB 8.0.13
Type or units	set of CPU IDs
Default	0
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

When used with `ndbd`, this parameter (now a string) specifies the ID of the CPU assigned to handle the `NDBCLUSTER` execution thread. When used with `ndbmtd`, the value of this parameter is a comma-separated list of CPU IDs assigned to handle execution threads. Each CPU ID in the list should be an integer in the range 0 to 65535 (inclusive).

The number of IDs specified should match the number of execution threads determined by `MaxNoOfExecutionThreads`. However, there is no guarantee that threads are assigned to CPUs

in any given order when using this parameter. You can obtain more finely-grained control of this type using `ThreadConfig`.

`LockExecuteThreadToCPU` has no default value.

- `LockMaintThreadsToCPU`

Version (or later)	NDB 8.0.13
Type or units	CPU ID
Default	0
Range	0 - 64K
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter specifies the ID of the CPU assigned to handle `NDBCLUSTER` maintenance threads.

The value of this parameter is an integer in the range 0 to 65535 (inclusive). *There is no default value*.

- `Numa`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	1
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter determines whether Non-Uniform Memory Access (NUMA) is controlled by the operating system or by the data node process, whether the data node uses `ndbd` or `ndbmtd`. By default, `NDB` attempts to use an interleaved NUMA memory allocation policy on any data node where the host operating system provides NUMA support.

Setting `Numa = 0` means that the datanode process does not itself attempt to set a policy for memory allocation, and permits this behavior to be determined by the operating system, which may be further guided by the separate `numactl` tool. That is, `Numa = 0` yields the system default behavior, which can be customised by `numactl`. For many Linux systems, the system default behavior is to allocate socket-local memory to any given process at allocation time. This can be problematic when using `ndbmtd`; this is because `nbdmtd` allocates all memory at startup, leading to an imbalance, giving different access speeds for different sockets, especially when locking pages in main memory.

Setting `Numa = 1` means that the data node process uses `libnuma` to request interleaved memory allocation. (This can also be accomplished manually, on the operating system level, using `numactl`.) Using interleaved allocation in effect tells the data node process to ignore non-uniform memory access but does not attempt to take any advantage of fast local memory; instead, the data node

process tries to avoid imbalances due to slow remote memory. If interleaved allocation is not desired, set `Numa` to 0 so that the desired behavior can be determined on the operating system level.

The `Numa` configuration parameter is supported only on Linux systems where `libnuma.so` is available.

- `RealtimeScheduler`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Setting this parameter to 1 enables real-time scheduling of data node threads.

The default is 0 (scheduling disabled).

- `SchedulerExecutionTimer`

Version (or later)	NDB 8.0.13
Type or units	µs
Default	50
Range	0 - 11000
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter specifies the time in microseconds for threads to be executed in the scheduler before being sent. Setting it to 0 minimizes the response time; to achieve higher throughput, you can increase the value at the expense of longer response times.

The default is 50 µsec, which our testing shows to increase throughput slightly in high-load cases without materially delaying requests.

- `SchedulerResponsiveness`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	5
Range	0 - 10
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

Set the balance in the `NDB` scheduler between speed and throughput. This parameter takes an integer whose value is in the range 0-10 inclusive, with 5 as the default. Higher values provide better response times relative to throughput. Lower values provide increased throughput at the expense of longer response times.

- `SchedulerSpinTimer`

Version (or later)	NDB 8.0.13
Type or units	µs
Default	0
Range	0 - 500
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter specifies the time in microseconds for threads to be executed in the scheduler before sleeping.

Starting with NDB 8.0.20, if `SpinMethod` is set, any setting for this parameter is ignored.

- `SpinMethod`

Version (or later)	NDB 8.0.20
Type or units	enumeration
Default	StaticSpinning
Range	CostBasedSpinning, LatencyOptimisedSpinning, DatabaseMachineSpinning, StaticSpinning
Added	NDB 8.0.20
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter is present beginning in NDB 8.0.20, but has no effect prior to NDB 8.0.24. It provides a simple interface to control adaptive spinning on data nodes, with four possible values furnishing presets for spin parameter values, as shown in the following list:

1. `StaticSpinning` (default): Sets `EnableAdaptiveSpinning` to `false` and `SchedulerSpinTimer` to 0. (`SetAllowedSpinOverhead` is not relevant in this case.)

2. `CostBasedSpinning`: Sets `EnableAdaptiveSpinning` to `true`, `SchedulerSpinTimer` to 100, and `SetAllowedSpinOverhead` to 200.

3. `LatencyOptimisedSpinning`: Sets `EnableAdaptiveSpinning` to `true`, `SchedulerSpinTimer` to 200, and `SetAllowedSpinOverhead` to 1000.

4. `DatabaseMachineSpinning`: Sets `EnableAdaptiveSpinning` to `true`, `SchedulerSpinTimer` to 500, and `SetAllowedSpinOverhead` to 10000. This is intended for use in cases where threads own their own CPUs.

The spin parameters modified by `SpinMethod` are described in the following list:

- `SchedulerSpinTimer`: This is the same as the data node configuration parameter of that name. The setting applied to this parameter by `SpinMethod` overrides any value set in the `config.ini` file.
- `EnableAdaptiveSpinning`: Enables or disables adaptive spinning. Disabling it causes spinning to be performed without making any checks for CPU resources. This parameter cannot be set directly in the cluster configuration file, and under most circumstances should not need to be, but can be enabled directly using `DUMP 104004 1` or disabled with `DUMP 104004 0` in the `ndb_mgm` management client.
- `SetAllowedSpinOverhead`: Sets the amount of CPU time to allow for gaining latency. This parameter cannot be set directly in the `config.ini` file. In most cases, the setting applied by SpinMethod should be satisfactory, but if it is necessary to change it directly, you can use `DUMP 104002 overhead` to do so, where *`overhead`* is a value ranging from 0 to 10000, inclusive; see the description of the indicated `DUMP` command for details.

On platforms lacking usable spin instructions, such as PowerPC and some SPARC platforms, spin time is set to 0 in all situations, and values for `SpinMethod` other than `StaticSpinning` are ignored.

- `TwoPassInitialNodeRestartCopy`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	true
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Multithreaded building of ordered indexes can be enabled for initial restarts of data nodes by setting this configuration parameter to `true` (the default value), which enables two-pass copying of data during initial node restarts.

You must also set `BuildIndexThreads` to a nonzero value.

Multi-Threading Configuration Parameters (ndbmtd). `ndbmtd` runs by default as a single-threaded process and must be configured to use multiple threads, using either of two methods, both of which require setting configuration parameters in the `config.ini` file. The first method is simply to set an appropriate value for the `MaxNoOfExecutionThreads` configuration parameter. A second method makes it possible to set up more complex rules for `ndbmtd` multithreading using `ThreadConfig`. The next few paragraphs provide information about these parameters and their use with multithreaded data nodes.

Note

A backup using parallelism on the data nodes requires that multiple LDMs are in use on all data nodes in the cluster prior to taking the backup. For

more information, see Section 23.6.8.5, “Taking an NDB Backup with Parallel Data Nodes”, as well as Section 23.5.23.3, “Restoring from a backup taken in parallel”.

- `AutomaticThreadConfig`

Version (or later)	NDB 8.0.23
Type or units	boolean
Default	false
Range	true, false
Added	NDB 8.0.23
Restart Type	**Initial System Restart:** Requires a complete shutdown of the cluster, wiping and restoring the cluster file system from a backup, and then restarting the cluster. (NDB 8.0.13)

When set to 1, enables automatic thread configuration employing the number of CPUs available to a data node taking into account any limits set by `taskset`, `numactl`, virtual machines, Docker, and other such means of controlling which CPUs are available to a given application (on Windows platforms, automatic thread configuration uses all CPUs which are online); alternatively, you can set `NumCPUs` to the desired number of CPUs (up to 1024, the maximum number of CPUs that can be handled by automatic thread configuration). Any settings for `ThreadConfig` and `MaxNoOfExecutionThreads` are ignored. In addition, enabling this parameter automatically disables `ClassicFragmentation`.

- `ClassicFragmentation`

Version (or later)	NDB 8.0.23
Type or units	boolean
Default	true
Range	true, false
Added	NDB 8.0.23
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

When enabled (set to `true`), `NDB` distributes fragments among LDMs in the manner always used by `NDB` prior to NDB 8.0.23; that is, the default number of partitions per node is equal to the minimum number of local data manager (LDM) threads per data node.

For new clusters for which a downgrade to NDB 8.0.22 or earlier is never expected to occur, setting `ClassicFragmentation` to `false` when first setting up the cluster is preferable; doing so causes

the number of partitions per node to be equal to the value of `PartitionsPerNode`, ensuring that all partitions are spread out evenly between all LDMs.

This parameter and `AutomaticThreadConfig` are mutually exclusive; enabling `AutomaticThreadConfig` automatically disables `ClassicFragmentation`.

- `EnableMultithreadedBackup`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	1
Range	0 - 1
Added	NDB 8.0.16
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Enables multi-threaded backup. If each data node has at least 2 LDMs, all LDM threads participate in the backup, which is created using one subdirectory per LDM thread, and each subdirectory containing `.ctl`, `.Data`, and `.log` backup files.

This parameter is normally enabled (set to 1) for `ndbmtd`. To force a single-threaded backup that can be restored easily using older versions of `ndb_restore`, disable multi-threaded backup by setting this parameter to 0. This must be done for each data node in the cluster.

See Section 23.6.8.5, “Taking an NDB Backup with Parallel Data Nodes”, and Section 23.5.23.3, “Restoring from a backup taken in parallel”, for more information.

- `MaxNoOfExecutionThreads`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	2
Range	2 - 72
Restart Type	**System Restart:** Requires a complete shutdown and restart of the cluster. (NDB 8.0.13)

This parameter directly controls the number of execution threads used by `ndbmtd`, up to a maximum of 72. Although this parameter is set in `[ndbd]` or `[ndbd default]` sections of the `config.ini` file, it is exclusive to `ndbmtd` and does not apply to `ndbd`.

Enabling `AutomaticThreadConfig` causes any setting for this parameter to be ignored.

Setting `MaxNoOfExecutionThreads` sets the number of threads for each type as determined by a matrix in the file `storage/ndb/src/common/mt_thr_config.cpp`. (Prior to NDB 8.0.30, this

was `storage/ndb/src/kernel/vm/mt_thr_config.cpp`.) This table shows these numbers of threads for possible values of `MaxNoOfExecutionThreads`.

Table 23.12 MaxNoOfExecutionThreads values and the corresponding number of threads by thread type (LQH, TC, Send, Receive).

`MaxNoOfExecutionThreads` Value	LDM Threads	TC Threads	Send Threads	Receive Threads
0 .. 3	1	0	0	1
4 .. 6	2	0	0	1
7 .. 8	4	0	0	1
9	4	2	0	1
10	4	2	1	1
11	4	3	1	1
12	6	2	1	1
13	6	3	1	1
14	6	3	1	2
15	6	3	2	2
16	8	3	1	2
17	8	4	1	2
18	8	4	2	2
19	8	5	2	2
20	10	4	2	2
21	10	5	2	2
22	10	5	2	3
23	10	6	2	3
24	12	5	2	3
25	12	6	2	3
26	12	6	3	3
27	12	7	3	3
28	12	7	3	4
29	12	8	3	4
30	12	8	4	4
31	12	9	4	4
32	16	8	3	3
33	16	8	3	4
34	16	8	4	4
35	16	9	4	4
36	16	10	4	4
37	16	10	4	5
38	16	11	4	5
39	16	11	5	5
40	20	10	4	4
41	20	10	4	5

MaxNoOfExecutionThreads Value	LDM Threads	TC Threads	Send Threads	Receive Threads
42	20	11	4	5
43	20	11	5	5
44	20	12	5	5
45	20	12	5	6
46	20	13	5	6
47	20	13	6	6
48	24	12	5	5
49	24	12	5	6
50	24	13	5	6
51	24	13	6	6
52	24	14	6	6
53	24	14	6	7
54	24	15	6	7
55	24	15	7	7
56	24	16	7	7
57	24	16	7	8
58	24	17	7	8
59	24	17	8	8
60	24	18	8	8
61	24	18	8	9
62	24	19	8	9
63	24	19	9	9
64	32	16	7	7
65	32	16	7	8
66	32	17	7	8
67	32	17	8	8
68	32	18	8	8
69	32	18	8	9
70	32	19	8	9
71	32	20	8	9
72	32	20	8	10

There is always one SUMA (replication) thread.

`NoOfFragmentLogParts` should be set equal to the number of LDM threads used by `ndbmtd`, as determined by the setting for this parameter. This ratio should not be any greater than 4:1; a configuration in which this is the case is specifically disallowed.

The number of LDM threads also determines the number of partitions used by an `NDB` table that is not explicitly partitioned; this is the number of LDM threads times the number of data nodes in the cluster. (If `ndbd` is used on the data nodes rather than `ndbmtd`, then there is always a single LDM thread; in this case, the number of partitions created automatically is simply equal to the number

of data nodes. See Section 23.2.2, “NDB Cluster Nodes, Node Groups, Fragment Replicas, and Partitions”, for more information.

Adding large tablespaces for Disk Data tables when using more than the default number of LDM threads may cause issues with resource and CPU usage if the disk page buffer is insufficiently large; see the description of the `DiskPageBufferMemory` configuration parameter, for more information.

The thread types are described later in this section (see `ThreadConfig`).

Setting this parameter outside the permitted range of values causes the management server to abort on startup with the error `Error line number: Illegal value value for parameter MaxNoOfExecutionThreads`.

For `MaxNoOfExecutionThreads`, a value of 0 or 1 is rounded up internally by `NDB` to 2, so that 2 is considered this parameter's default and minimum value.

`MaxNoOfExecutionThreads` is generally intended to be set equal to the number of CPU threads available, and to allocate a number of threads of each type suitable to typical workloads. It does not assign particular threads to specified CPUs. For cases where it is desirable to vary from the settings provided, or to bind threads to CPUs, you should use `ThreadConfig` instead, which allows you to allocate each thread directly to a desired type, CPU, or both.

The multithreaded data node process always spawns, at a minimum, the threads listed here:

- 1 local query handler (LDM) thread
- 1 receive thread
- 1 subscription manager (SUMA or replication) thread

For a `MaxNoOfExecutionThreads` value of 8 or less, no TC threads are created, and TC handling is instead performed by the main thread.

Changing the number of LDM threads normally requires a system restart, whether it is changed using this parameter or `ThreadConfig`, but it is possible to effect the change using a node initial restart (*NI*) provided the following two conditions are met:

- Each LDM thread handles a maximum of 8 fragments, and
- The total number of table fragments is an integer multiple of the number of LDM threads.

In NDB 8.0, an initial restart is *not* required to effect a change in this parameter, as it was in some older versions of NDB Cluster.

- `NoOfFragmentLogParts`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	4
Range	4, 6, 8, 10, 12, 16, 20, 24, 32
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be

	restarted with `--initial`. (NDB 8.0.13)

Set the number of log file groups for redo logs belonging to this `ndbmtd`. The value of this parameter should be set equal to the number of LDM threads used by `ndbmtd` as determined by the setting for `MaxNoOfExecutionThreads`. A configuration using more than 4 redo log parts per LDM is disallowed.

See the description of `MaxNoOfExecutionThreads` for more information.

- `NumCPUs`

Version (or later)	NDB 8.0.23
Type or units	integer
Default	0
Range	0 - 1024
Added	NDB 8.0.23
Restart Type	**Initial System Restart:** Requires a complete shutdown of the cluster, wiping and restoring the cluster file system from a backup, and then restarting the cluster. (NDB 8.0.13)

Cause automatic thread configuration to use only this many CPUs. Has no effect if `AutomaticThreadConfig` is not enabled.

- `PartitionsPerNode`

Version (or later)	NDB 8.0.23
Type or units	integer
Default	2
Range	1 - 32
Added	NDB 8.0.23
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

Sets the number of partitions used on each node when creating a new `NDB` table. This makes it possible to avoid splitting up tables into an excessive number of partitions when the number of local data managers (LDMs) grows high.

While it is possible to set this parameter to different values on different data nodes and there are no known issues with doing so, this is also not likely to be of any advantage; for this reason, it is recommended simply to set it once, for all data nodes, in the `[ndbd default]` section of the global `config.ini` file.

If `ClassicFragmentation` is enabled, any setting for this parameter is ignored. (Remember that enabling `AutomaticThreadConfig` disables `ClassicFragmentation`.)

- `ThreadConfig`

Version (or later)	NDB 8.0.13
Type or units	string
Default	''
Range	...
Restart Type	**System Restart:** Requires a complete shutdown and restart of the cluster. (NDB 8.0.13)

This parameter is used with `ndbmtd` to assign threads of different types to different CPUs. Its value is a string whose format has the following syntax:

```
ThreadConfig := entry[,entry[,...]]

entry := type={param[,param[,...]]}

type (NDB 8.0.22 and earlier) := ldm | main | recv | send | rep | io | tc | watchdog | idxbld

type (NDB 8.0.23 and later) := ldm | query | recover | main | recv | send | rep | io | tc | watchdog | io

param := count=number
  | cpubind=cpu_list
  | cpuset=cpu_list
  | spintime=number
  | realtime={0|1}
  | nosend={0|1}
  | thread_prio={0..10}
  | cpubind_exclusive=cpu_list
  | cpuset_exclusive=cpu_list
```

The curly braces ({...}) surrounding the list of parameters are required, even if there is only one parameter in the list.

A *`param`* (parameter) specifies any or all of the following information:

- The number of threads of the given type (`count`).
- The set of CPUs to which the threads of the given type are to be nonexclusively bound. This is determined by either one of `cpubind` or `cpuset`). `cpubind` causes each thread to be bound

(nonexclusively) to a CPU in the set; `cpuset` means that each thread is bound (nonexclusively) to the set of CPUs specified.

On Solaris, you can instead specify a set of CPUs to which the threads of the given type are to be bound exclusively. `cpubind_exclusive` causes each thread to be bound exclusively to a CPU in the set; `cpuset_exclsuive` means that each thread is bound exclusively to the set of CPUs specified.

Only one of `cpubind`, `cpuset`, `cpubind_exclusive`, or `cpuset_exclusive` can be provided in a single configuration.

- `spintime` determines the wait time in microseconds the thread spins before going to sleep.

  The default value for `spintime` is the value of the `SchedulerSpinTimer` data node configuration parameter.

  `spintime` does not apply to I/O threads, watchdog, or offline index build threads, and so cannot be set for these thread types.

- `realtime` can be set to 0 or 1. If it is set to 1, the threads run with real-time priority. This also means that `thread_prio` cannot be set.

  The `realtime` parameter is set by default to the value of the `RealtimeScheduler` data node configuration parameter.

  `realtime` cannot be set for offline index build threads.

- By setting `nosend` to 1, you can prevent a `main`, `ldm`, `rep`, or `tc` thread from assisting the send threads. This parameter is 0 by default, and cannot be used with other types of threads.

- `thread_prio` is a thread priority level that can be set from 0 to 10, with 10 representing the greatest priority. The default is 5. The precise effects of this parameter are platform-specific, and are described later in this section.

  The thread priority level cannot be set for offline index build threads.

thread_prio settings and effects by platform. The implementation of `thread_prio` differs between Linux/FreeBSD, Solaris, and Windows. In the following list, we discuss its effects on each of these platforms in turn:

- *Linux and FreeBSD*: We map `thread_prio` to a value to be supplied to the `nice` system call. Since a lower niceness value for a process indicates a higher process priority, increasing `thread_prio` has the effect of lowering the `nice` value.

  Table 23.13 Mapping of thread_prio to nice values on Linux and FreeBSD

`thread_prio` value	`nice` value
0	19
1	16
2	12
3	8
4	4
5	0
6	-4
7	-8
8	-12

thread_prio value	nice value
9	-16
10	-20

Some operating systems may provide for a maximum process niceness level of 20, but this is not supported by all targeted versions; for this reason, we choose 19 as the maximum `nice` value that can be set.

- *Solaris*: Setting `thread_prio` on Solaris sets the Solaris FX priority, with mappings as shown in the following table:

Table 23.14 Mapping of thread_prio to FX priority on Solaris

thread_prio value	Solaris FX priority
0	15
1	20
2	25
3	30
4	35
5	40
6	45
7	50
8	55
9	59
10	60

A `thread_prio` setting of 9 is mapped on Solaris to the special FX priority value 59, which means that the operating system also attempts to force the thread to run alone on its own CPU core.

- *Windows*: We map `thread_prio` to a Windows thread priority value passed to the Windows API `SetThreadPriority()` function. This mapping is shown in the following table:

Table 23.15 Mapping of thread_prio to Windows thread priority

thread_prio value	Windows thread priority
0 - 1	THREAD_PRIORITY_LOWEST
2 - 3	THREAD_PRIORITY_BELOW_NORMAL
4 - 5	THREAD_PRIORITY_NORMAL
6 - 7	THREAD_PRIORITY_ABOVE_NORMAL
8 - 10	THREAD_PRIORITY_HIGHEST

The *`type`* attribute represents an NDB thread type. The thread types supported, and the range of permitted `count` values for each, are provided in the following list:

- `ldm`: Local query handler (`DBLQH` kernel block) that handles data. The more LDM threads that are used, the more highly partitioned the data becomes. (Beginning with NDB 8.0.23, when `ClassicFragmentation` is set to 0, the number of partitions is independent of the number of LDM threads, and depends on the value of `PartitionsPerNode` instead.) Each LDM thread maintains its own sets of data and index partitions, as well as its own redo log. Prior to NDB 8.0.23, the value set for `ldm` must be one of the values 1, 2, 4, 6, 8, 12, 16, 24, or 32. In NDB 8.0.23 and later, it is possible to set `ldm` to any value in the range 1 to 332 inclusive; it also

becomes possible to set it to 0, provided that `main`, `rep`, and `tc` are also 0, and that `recv` is set to 1; doing this causes `ndbmtd` to emulate `ndbd`.

Each LDM thread is normally grouped with 1 query thread to form an LDM group. A set of 4 to 8 LDM groups is grouped into a round robin groups. Each LDM thread can be assisted in execution by any query or threads in the same round robin group. `NDB` attempts to form round robin groups such that all threads in each round robin group are locked to CPUs that are attached to the same L3 cache, within the limits of the range stated for a round robin group's size.

Changing the number of LDM threads normally requires a system restart to be effective and safe for cluster operations; this requirement is relaxed in certain cases, as explained later in this section. This is also true when this is done using `MaxNoOfExecutionThreads`.

Adding large tablespaces (hundreds of gigabytes or more) for Disk Data tables when using more than the default number of LDMs may cause issues with resource and CPU usage if `DiskPageBufferMemory` is not sufficiently large.

In NDB 8.0.30 (only), `ldm` must be included in the `ThreadConfig` value string. Beginning with NDB 8.0.31, if this is omitted, one `ldm` thread is created. These changes may affect upgrades from previous releases; see Section 23.3.7, “Upgrading and Downgrading NDB Cluster”, for more information.

- `query` (Added in NDB 8.0.23): A query thread is tied to an LDM and together with it forms an LDM group; acts only on `READ COMMITTED` queries. The number of query threads must be set to 0, 1, 2, or 3 times the number of LDM threads. Query threads are not used, unless this is overridden by setting `query` to a nonzero value, or by enabling the `AutomaticThreadConfig` parameter, in which case LDMs behave as they did prior to NDB 8.0.23.

  A query thread also acts as a recovery thread (see next item), although the reverse is not true.

  Changing the number of query threads requires a node restart.

- `recover` (Added in NDB 8.0.23): A recovery thread restores data from a fragment as part of an LCP.

  Changing the number of recovery threads requires a node restart.

- `tc`: Transaction coordinator thread (`DBTC` kernel block) containing the state of an ongoing transaction. In NDB 8.0.23 and later, the maximum number of TC threads is 128; previously, this was 32.

  Optimally, every new transaction can be assigned to a new TC thread. In most cases 1 TC thread per 2 LDM threads is sufficient to guarantee that this can happen. In cases where the number of writes is relatively small when compared to the number of reads, it is possible that only 1 TC thread per 4 LQH threads is required to maintain transaction states. Conversely, in applications that perform a great many updates, it may be necessary for the ratio of TC threads to LDM threads to approach 1 (for example, 3 TC threads to 4 LDM threads).

  Setting `tc` to 0 causes TC handling to be done by the main thread. In most cases, this is effectively the same as setting it to 1.

  Range: 0-64 (*NDB 8.0.22 and earlier*: 0 - 32)

- `main`: Data dictionary and transaction coordinator (`DBDIH` and `DBTC` kernel blocks), providing schema management. Prior to NDB 8.0.23, this was always handled by a single dedicated thread, beginning with NDB 8.0.23, it is also possible to specify zero or two main threads.

  Range:

  - *NDB 8.0.22 and earlier*: 1 only.

    NDB 8.0.23 and later: 0-2.

    Setting `main` to 0 and `rep` to 1 causes the `main` blocks to be placed into the `rep` thread; the combined thread is shown in the `ndbinfo.threads` table as `main_rep`. This is effectively the same as setting `rep` equal to 1 and `main` equal to 0.

    It is also possible to set both `main` and `rep` to 0, in which case both threads are placed in the first `recv` thread; the resulting combined thread is named `main_rep_recv` in the `threads` table.

  In NDB 8.0.30 (only), `main` must be included in the `ThreadConfig` value string. Beginning with NDB 8.0.31, if this is omitted, one `main` thread is created. These changes may affect upgrades from previous releases; see Section 23.3.7, “Upgrading and Downgrading NDB Cluster”, for more information.

- `recv`: Receive thread (`CMVMI` kernel block). Each receive thread handles one or more sockets for communicating with other nodes in an NDB Cluster, with one socket per node. NDB Cluster supports multiple receive threads; the maximum is 16 such threads.

  Range:

  - *NDB 8.0.22 and earlier*: 1 - 16
  - *NDB 8.0.23 and later*: 1 - 64

  In NDB 8.0.30 (only), `recv` must be included in the `ThreadConfig` value string. Beginning with NDB 8.0.31, if this is omitted, one `recv` thread is created. These changes may affect upgrades from previous releases; see Section 23.3.7, “Upgrading and Downgrading NDB Cluster”, for more information.

- `send`: Send thread (`CMVMI` kernel block). To increase throughput, it is possible to perform sends from one or more separate, dedicated threads (maximum 8).

  In NDB 8.0.20 and later, due to changes in the multithreading implementation, using many send threads can have an adverse effect on scalability.

  Previously, all threads handled their own sending directly; this can still be made to happen by setting the number of send threads to 0 (this also happens when `MaxNoOfExecutionThreads` is set less than 10). While doing so can have an adverse impact on throughput, it can also in some cases provide decreased latency.

  Range:

  - *NDB 8.0.22 and earlier*: 0 - 16
  - *NDB 8.0.23 and later*: 0 - 64

- `rep`: Replication thread (`SUMA` kernel block). Prior to NDB 8.0.23, asynchronous replication operations are always handled by a single, dedicated thread. Beginning with NDB 8.0.23, this thread can be combined with the main thread (see range information).

  Range:

  - *NDB 8.0.22 and earlier*: 1 only.
  - *NDB 8.0.23 and later*: 0-1.

    Setting `rep` to 0 and `main` to 1 causes the `rep` blocks to be placed into the `main` thread; the combined thread is shown in the `ndbinfo.threads` table as `main_rep`. This is effectively the same as setting `main` equal to 1 and `rep` equal to 0.

    It is also possible to set both `main` and `rep` to 0, in which case both threads are placed in the first `recv` thread; the resulting combined thread is named `main_rep_recv` in the `threads` table.

  In NDB 8.0.30 (only), `rep` must be included in the `ThreadConfig` value string. Beginning with NDB 8.0.31, if this is omitted, one `rep` thread is created. These changes may affect upgrades from previous releases; see Section 23.3.7, “Upgrading and Downgrading NDB Cluster”, for more information.

- `io`: File system and other miscellaneous operations. These are not demanding tasks, and are always handled as a group by a single, dedicated I/O thread.

  Range: 1 only.

- `watchdog`: Parameters settings associated with this type are actually applied to several threads, each having a specific use. These threads include the `SocketServer` thread, which receives connection setups from other nodes; the `SocketClient` thread, which attempts to set up connections to other nodes; and the thread watchdog thread that checks that threads are progressing.

  Range: 1 only.

- `idxbld`: Offline index build threads. Unlike the other thread types listed previously, which are permanent, these are temporary threads which are created and used only during node or system restarts, or when running `ndb_restore --rebuild-indexes`. They may be bound to CPU sets which overlap with CPU sets bound to permanent thread types.

  `thread_prio`, `realtime`, and `spintime` values cannot be set for offline index build threads. In addition, `count` is ignored for this type of thread.

  If `idxbld` is not specified, the default behavior is as follows:

  - Offline index build threads are not bound if the I/O thread is also not bound, and these threads use any available cores.
  - If the I/O thread is bound, then the offline index build threads are bound to the entire set of bound threads, due to the fact that there should be no other tasks for these threads to perform.

  Range: 0 - 1.

Changing `ThreadCOnfig` normally requires a system initial restart, but this requirement can be relaxed under certain circumstances:

- If, following the change, the number of LDM threads remains the same as before, nothing more than a simple node restart (rolling restart, or *N*) is required to implement the change.

- Otherwise (that is, if the number of LDM threads changes), it is still possible to effect the change using a node initial restart (*NI*) provided the following two conditions are met:

  a. Each LDM thread handles a maximum of 8 fragments, and

  b. The total number of table fragments is an integer multiple of the number of LDM threads.

In any other case, a system initial restart is needed to change this parameter.

`NDB` can distinguish between thread types by both of the following criteria:

- Whether the thread is an execution thread. Threads of type `main`, `ldm`, `query` (NDB 8.0.23 and later), `recv`, `rep`, `tc`, and `send` are execution threads; `io`, `recover` (NDB 8.0.23 and later), `watchdog`, and `idxbld` threads are not considered execution threads.
- Whether the allocation of threads to a given task is permanent or temporary. Currently all thread types except `idxbld` are considered permanent; `idxbld` threads are regarded as temporary threads.

Simple examples:

```
# Example 1.

ThreadConfig=ldm={count=2,cpubind=1,2},main={cpubind=12},rep={cpubind=11}

# Example 2.

Threadconfig=main={cpubind=0},ldm={count=4,cpubind=1,2,5,6},io={cpubind=3}
```

It is usually desirable when configuring thread usage for a data node host to reserve one or more number of CPUs for operating system and other tasks. Thus, for a host machine with 24 CPUs, you might want to use 20 CPU threads (leaving 4 for other uses), with 8 LDM threads, 4 TC threads (half the number of LDM threads), 3 send threads, 3 receive threads, and 1 thread each for schema management, asynchronous replication, and I/O operations. (This is almost the same distribution of threads used when `MaxNoOfExecutionThreads` is set equal to 20.) The following `ThreadConfig` setting performs these assignments, additionally binding all of these threads to specific CPUs:

```
ThreadConfig=ldm{count=8,cpubind=1,2,3,4,5,6,7,8},main={cpubind=9},io={cpubind=9}, \
rep={cpubind=10},tc{count=4,cpubind=11,12,13,14},recv={count=3,cpubind=15,16,17}, \
send{count=3,cpubind=18,19,20}
```

It should be possible in most cases to bind the main (schema management) thread and the I/O thread to the same CPU, as we have done in the example just shown.

The following example incorporates groups of CPUs defined using both `cpuset` and `cpubind`, as well as use of thread prioritization.

```
ThreadConfig=ldm={count=4,cpuset=0-3,thread_prio=8,spintime=200}, \
ldm={count=4,cpubind=4-7,thread_prio=8,spintime=200}, \
tc={count=4,cpuset=8-9,thread_prio=6},send={count=2,thread_prio=10,cpubind=10-11}, \
main={count=1,cpubind=10},rep={count=1,cpubind=11}
```

In this case we create two LDM groups; the first uses `cpubind` and the second uses `cpuset`. `thread_prio` and `spintime` are set to the same values for each group. This means there are eight LDM threads in total. (You should ensure that `NoOfFragmentLogParts` is also set to 8.) The four TC threads use only two CPUs; it is possible when using `cpuset` to specify fewer CPUs than threads in the group. (This is not true for `cpubind`.) The send threads use two threads using

`cpubind` to bind these threads to CPUs 10 and 11. The main and rep threads can reuse these CPUs.

This example shows how `ThreadConfig` and `NoOfFragmentLogParts` might be set up for a 24-CPU host with hyperthreading, leaving CPUs 10, 11, 22, and 23 available for operating system functions and interrupts:

```
NoOfFragmentLogParts=10
ThreadConfig=ldm={count=10,cpubind=0-4,12-16,thread_prio=9,spintime=200}, \
tc={count=4,cpuset=6-7,18-19,thread_prio=8},send={count=1,cpuset=8}, \
recv={count=1,cpuset=20},main={count=1,cpuset=9,21},rep={count=1,cpuset=9,21}, \
io={count=1,cpuset=9,21,thread_prio=8},watchdog={count=1,cpuset=9,21,thread_prio=9}
```

The next few examples include settings for `idxbld`. The first two of these demonstrate how a CPU set defined for `idxbld` can overlap those specified for other (permanent) thread types, the first using `cpuset` and the second using `cpubind`:

```
ThreadConfig=main,ldm={count=4,cpuset=1-4},tc={count=4,cpuset=5,6,7}, \
io={cpubind=8},idxbld={cpuset=1-8}

ThreadConfig=main,ldm={count=1,cpubind=1},idxbld={count=1,cpubind=1}
```

The next example specifies a CPU for the I/O thread, but not for the index build threads:

```
ThreadConfig=main,ldm={count=4,cpuset=1-4},tc={count=4,cpuset=5,6,7}, \
io={cpubind=8}
```

Since the `ThreadConfig` setting just shown locks threads to eight cores numbered 1 through 8, it is equivalent to the setting shown here:

```
ThreadConfig=main,ldm={count=4,cpuset=1-4},tc={count=4,cpuset=5,6,7}, \
io={cpubind=8},idxbld={cpuset=1,2,3,4,5,6,7,8}
```

In order to take advantage of the enhanced stability that the use of `ThreadConfig` offers, it is necessary to insure that CPUs are isolated, and that they not subject to interrupts, or to being scheduled for other tasks by the operating system. On many Linux systems, you can do this by setting `IRQBALANCE_BANNED_CPUS` in `/etc/sysconfig/irqbalance` to `0xFFFFF0`, and by using the `isolcpus` boot option in `grub.conf`. For specific information, see your operating system or platform documentation.

Disk Data Configuration Parameters. Configuration parameters affecting Disk Data behavior include the following:

- `DiskPageBufferEntries`

Version (or later)	NDB 8.0.13
Type or units	32K pages
Default	10
Range	1 - 1000
Version (or later)	NDB 8.0.19
Type or units	bytes
Default	64MB
Range	4MB - 16TB
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

This is the number of page entries (page references) to allocate. It is specified as a number of 32K pages in `DiskPageBufferMemory`. The default is sufficient for most cases but you may need to increase the value of this parameter if you encounter problems with very large transactions on Disk Data tables. Each page entry requires approximately 100 bytes.

- `DiskPageBufferMemory`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	64M
Range	4M - 1T
Version (or later)	NDB 8.0.19
Type or units	bytes
Default	64M
Range	4M - 16T
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This determines the amount of space used for caching pages on disk, and is set in the `[ndbd]` or `[ndbd default]` section of the `config.ini` file.

Note

Previously, this parameter was specified as a number of 32 KB pages. In NDB 8.0, it is specified as a number of bytes.

If the value for `DiskPageBufferMemory` is set too low in conjunction with using more than the default number of LDM threads in `ThreadConfig` (for example `{ldm=6...}`), problems can arise when trying to add a large (for example 500G) data file to a disk-based `NDB` table, wherein the process takes indefinitely long while occupying one of the CPU cores.

This is due to the fact that, as part of adding a data file to a tablespace, extent pages are locked into memory in an extra PGMAN worker thread, for quick metadata access. When adding a large file, this worker has insufficient memory for all of the data file metadata. In such cases, you should either increase `DiskPageBufferMemory`, or add smaller tablespace files. You may also need to adjust `DiskPageBufferEntries`.

You can query the `ndbinfo.diskpagebuffer` table to help determine whether the value for this parameter should be increased to minimize unnecessary disk seeks. See Section 23.6.16.30, “The ndbinfo diskpagebuffer Table”, for more information.

- `SharedGlobalMemory`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	128M

Range	0 - 64T
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter determines the amount of memory that is used for log buffers, disk operations (such as page requests and wait queues), and metadata for tablespaces, log file groups, `UNDO` files, and data files. The shared global memory pool also provides memory used for satisfying the memory requirements of the `UNDO_BUFFER_SIZE` option used with `CREATE LOGFILE GROUP` and `ALTER LOGFILE GROUP` statements, including any default value implied for this options by the setting of the `InitialLogFileGroup` data node configuration parameter. `SharedGlobalMemory` can be set in the `[ndbd]` or `[ndbd default]` section of the `config.ini` configuration file, and is measured in bytes.

The default value is `128M`.

- `DiskIOThreadPool`

Version (or later)	NDB 8.0.13
Type or units	threads
Default	2
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter determines the number of unbound threads used for Disk Data file access. Before `DiskIOThreadPool` was introduced, exactly one thread was spawned for each Disk Data file, which could lead to performance issues, particularly when using very large data files. With `DiskIOThreadPool`, you can—for example—access a single large data file using several threads working in parallel.

This parameter applies to Disk Data I/O threads only.

The optimum value for this parameter depends on your hardware and configuration, and includes these factors:

- **Physical distribution of Disk Data files.** You can obtain better performance by placing data files, undo log files, and the data node file system on separate physical disks. If you do this with some or all of these sets of files, then you can (and should) set `DiskIOThreadPool` higher to enable separate threads to handle the files on each disk.

  In NDB 8.0, you should also disable `DiskDataUsingSameDisk` when using a separate disk or disks for Disk Data files; this increases the rate at which checkpoints of Disk Data tablespaces can be performed.

- **Disk performance and types.** The number of threads that can be accommodated for Disk Data file handling is also dependent on the speed and throughput of the disks. Faster disks and higher throughput allow for more disk I/O threads. Our test results indicate that solid-state disk

drives can handle many more disk I/O threads than conventional disks, and thus higher values for `DiskIOThreadPool`.

Decreasing `TimeBetweenGlobalCheckpoints` is also recommended when using solid-state disk drives, in particular those using NVMe. See also Disk Data latency parameters.

The default value for this parameter is 2.

- **Disk Data file system parameters.** The parameters in the following list make it possible to place NDB Cluster Disk Data files in specific directories without the need for using symbolic links.

  - `FileSystemPathDD`

Version (or later)	NDB 8.0.13
Type or units	filename
Default	FileSystemPath
Range	...
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be restarted with `--initial`. (NDB 8.0.13)

If this parameter is specified, then NDB Cluster Disk Data data files and undo log files are placed in the indicated directory. This can be overridden for data files, undo log files, or both, by specifying values for `FileSystemPathDataFiles`, `FileSystemPathUndoFiles`, or both, as explained for these parameters. It can also be overridden for data files by specifying a path in the `ADD DATAFILE` clause of a `CREATE TABLESPACE` or `ALTER TABLESPACE` statement, and for undo log files by specifying a path in the `ADD UNDOFILE` clause of a `CREATE LOGFILE GROUP` or `ALTER LOGFILE GROUP` statement. If `FileSystemPathDD` is not specified, then `FileSystemPath` is used.

If a `FileSystemPathDD` directory is specified for a given data node (including the case where the parameter is specified in the `[ndbd default]` section of the `config.ini` file), then starting that data node with `--initial` causes all files in the directory to be deleted.

  - `FileSystemPathDataFiles`

Version (or later)	NDB 8.0.13
Type or units	filename
Default	FileSystemPathDD
Range	...
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be

	restarted with `--initial`. (NDB 8.0.13)

If this parameter is specified, then NDB Cluster Disk Data data files are placed in the indicated directory. This overrides any value set for `FileSystemPathDD`. This parameter can be overridden for a given data file by specifying a path in the `ADD DATAFILE` clause of a `CREATE TABLESPACE` or `ALTER TABLESPACE` statement used to create that data file. If `FileSystemPathDataFiles` is not specified, then `FileSystemPathDD` is used (or `FileSystemPath`, if `FileSystemPathDD` has also not been set).

If a `FileSystemPathDataFiles` directory is specified for a given data node (including the case where the parameter is specified in the `[ndbd default]` section of the `config.ini` file), then starting that data node with `--initial` causes all files in the directory to be deleted.

- `FileSystemPathUndoFiles`

Version (or later)	NDB 8.0.13
Type or units	filename
Default	FileSystemPathDD
Range	...
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be restarted with `--initial`. (NDB 8.0.13)

If this parameter is specified, then NDB Cluster Disk Data undo log files are placed in the indicated directory. This overrides any value set for `FileSystemPathDD`. This parameter can be overridden for a given data file by specifying a path in the `ADD UNDO` clause of a `CREATE LOGFILE GROUP` or `ALTER LOGFILE GROUP` statement used to create that data file. If `FileSystemPathUndoFiles` is not specified, then `FileSystemPathDD` is used (or `FileSystemPath`, if `FileSystemPathDD` has also not been set).

If a `FileSystemPathUndoFiles` directory is specified for a given data node (including the case where the parameter is specified in the `[ndbd default]` section of the `config.ini` file), then starting that data node with `--initial` causes all files in the directory to be deleted.

For more information, see Section 23.6.11.1, “NDB Cluster Disk Data Objects”.

- **Disk Data object creation parameters.** The next two parameters enable you—when starting the cluster for the first time—to cause a Disk Data log file group, tablespace, or both, to be created without the use of SQL statements.

  - `InitialLogFileGroup`

Version (or later)	NDB 8.0.13
Type or units	string

Default	[see documentation]
Range	...
Restart Type	**System Restart:** Requires a complete shutdown and restart of the cluster. (NDB 8.0.13)

This parameter can be used to specify a log file group that is created when performing an initial start of the cluster. `InitialLogFileGroup` is specified as shown here:

```
InitialLogFileGroup = [name=name;] [undo_buffer_size=size;] file-specification-list

file-specification-list:
    file-specification[; file-specification[; ...]]

file-specification:
    filename:size
```

The `name` of the log file group is optional and defaults to `DEFAULT-LG`. The `undo_buffer_size` is also optional; if omitted, it defaults to `64M`. Each *`file-specification`* corresponds to an undo log file, and at least one must be specified in the *`file-specification-list`*. Undo log files are placed according to any values that have been set for `FileSystemPath`, `FileSystemPathDD`, and `FileSystemPathUndoFiles`, just as if they had been created as the result of a `CREATE LOGFILE GROUP` or `ALTER LOGFILE GROUP` statement.

Consider the following:

```
InitialLogFileGroup = name=LG1; undo_buffer_size=128M; undo1.log:250M; undo2.log:150M
```

This is equivalent to the following SQL statements:

```
CREATE LOGFILE GROUP LG1
    ADD UNDOFILE 'undo1.log'
    INITIAL_SIZE 250M
    UNDO_BUFFER_SIZE 128M
    ENGINE NDBCLUSTER;

ALTER LOGFILE GROUP LG1
    ADD UNDOFILE 'undo2.log'
    INITIAL_SIZE 150M
    ENGINE NDBCLUSTER;
```

This logfile group is created when the data nodes are started with `--initial`.

Resources for the initial log file group are added to the global memory pool along with those indicated by the value of `SharedGlobalMemory`.

This parameter, if used, should always be set in the `[ndbd default]` section of the `config.ini` file. The behavior of an NDB Cluster when different values are set on different data nodes is not defined.

- `InitialTablespace`

Version (or later)	NDB 8.0.13
Type or units	string

Default	[see documentation]
Range	...
Restart Type	**System Restart:** Requires a complete shutdown and restart of the cluster. (NDB 8.0.13)

This parameter can be used to specify an NDB Cluster Disk Data tablespace that is created when performing an initial start of the cluster. `InitialTablespace` is specified as shown here:

```
InitialTablespace = [name=name;] [extent_size=size;] file-specification-list
```

The `name` of the tablespace is optional and defaults to `DEFAULT-TS`. The `extent_size` is also optional; it defaults to `1M`. The *`file-specification-list`* uses the same syntax as shown with the `InitialLogfileGroup` parameter, the only difference being that each *`file-specification`* used with `InitialTablespace` corresponds to a data file. At least one must be specified in the *`file-specification-list`*. Data files are placed according to any values that have been set for `FileSystemPath`, `FileSystemPathDD`, and `FileSystemPathDataFiles`, just as if they had been created as the result of a `CREATE TABLESPACE` or `ALTER TABLESPACE` statement.

For example, consider the following line specifying `InitialTablespace` in the `[ndbd default]` section of the `config.ini` file (as with `InitialLogfileGroup`, this parameter should always be set in the `[ndbd default]` section, as the behavior of an NDB Cluster when different values are set on different data nodes is not defined):

```
InitialTablespace = name=TS1; extent_size=8M; data1.dat:2G; data2.dat:4G
```

This is equivalent to the following SQL statements:

```
CREATE TABLESPACE TS1
    ADD DATAFILE 'data1.dat'
    EXTENT_SIZE 8M
    INITIAL_SIZE 2G
    ENGINE NDBCLUSTER;

ALTER TABLESPACE TS1
    ADD DATAFILE 'data2.dat'
    INITIAL_SIZE 4G
    ENGINE NDBCLUSTER;
```

This tablespace is created when the data nodes are started with `--initial`, and can be used whenever creating NDB Cluster Disk Data tables thereafter.

- **Disk Data latency parameters.** The two parameters listed here can be used to improve handling of latency issues with NDB Cluster Disk Data tables.

  - `MaxDiskDataLatency`

Version (or later)	NDB 8.0.19
Type or units	ms
Default	0
Range	0 - 8000

Added	NDB 8.0.19
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter controls the maximum allowed mean latency for disk access (maximum 8000 milliseconds). When this limit is reached, `NDB` begins to abort transactions in order to decrease pressure on the Disk Data I/O subsystem. Use `0` to disable the latency check.

- `DiskDataUsingSameDisk`

Version (or later)	NDB 8.0.19
Type or units	boolean
Default	true
Range	...
Added	NDB 8.0.19
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Set this parameter to `false` if your Disk Data tablespaces use one or more separate disks. Doing so allows checkpoints to tablespaces to be executed at a higher rate than normally used for when disks are shared.

When `DiskDataUsingSameDisk` is `true`, `NDB` decreases the rate of Disk Data checkpointing whenever an in-memory checkpoint is in progress to help ensure that disk load remains constant.

Disk Data and GCP Stop errors. Errors encountered when using Disk Data tables such as `Node *nodeid* killed this node because GCP stop was detected` (error 2303) are often referred to as “GCP stop errors”. Such errors occur when the redo log is not flushed to disk quickly enough; this is usually due to slow disks and insufficient disk throughput.

You can help prevent these errors from occurring by using faster disks, and by placing Disk Data files on a separate disk from the data node file system. Reducing the value of `TimeBetweenGlobalCheckpoints` tends to decrease the amount of data to be written for each global checkpoint, and so may provide some protection against redo log buffer overflows when trying to write a global checkpoint; however, reducing this value also permits less time in which to write the GCP, so this must be done with caution.

In addition to the considerations given for `DiskPageBufferMemory` as explained previously, it is also very important that the `DiskIOThreadPool` configuration parameter be set correctly; having `DiskIOThreadPool` set too high is very likely to cause GCP stop errors (Bug #37227).

GCP stops can be caused by save or commit timeouts; the `TimeBetweenEpochsTimeout` data node configuration parameter determines the timeout for commits. However, it is possible to disable both types of timeouts by setting this parameter to 0.

Parameters for configuring send buffer memory allocation. Send buffer memory is allocated dynamically from a memory pool shared between all transporters, which means that the size of the send buffer can be adjusted as necessary. (Previously, the NDB kernel used a fixed-size send buffer for every node in the cluster, which was allocated when the node started and could not be changed

while the node was running.) The `TotalSendBufferMemory` and `OverLoadLimit` data node configuration parameters permit the setting of limits on this memory allocation. For more information about the use of these parameters (as well as `SendBufferMemory`), see Section 23.4.3.14, “Configuring NDB Cluster Send Buffer Parameters”.

- `ExtraSendBufferMemory`

  This parameter specifies the amount of transporter send buffer memory to allocate in addition to any set using `TotalSendBufferMemory`, `SendBufferMemory`, or both.

- `TotalSendBufferMemory`

  This parameter is used to determine the total amount of memory to allocate on this node for shared send buffer memory among all configured transporters.

  If this parameter is set, its minimum permitted value is 256KB; 0 indicates that the parameter has not been set. For more detailed information, see Section 23.4.3.14, “Configuring NDB Cluster Send Buffer Parameters”.

See also Section 23.6.7, “Adding NDB Cluster Data Nodes Online”.

Redo log over-commit handling. It is possible to control a data node's handling of operations when too much time is taken flushing redo logs to disk. This occurs when a given redo log flush takes longer than `RedoOverCommitLimit` seconds, more than `RedoOverCommitCounter` times, causing any pending transactions to be aborted. When this happens, the API node that sent the transaction can handle the operations that should have been committed either by queuing the operations and re-trying them, or by aborting them, as determined by `DefaultOperationRedoProblemAction`. The data node configuration parameters for setting the timeout and number of times it may be exceeded before the API node takes this action are described in the following list:

- `RedoOverCommitCounter`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	3
Range	1 - 4294967039 (0xFFFFFEFF)
Version (or later)	NDB 8.0.19
Type or units	numeric
Default	3
Range	1 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

  When `RedoOverCommitLimit` is exceeded when trying to write a given redo log to disk this many times or more, any transactions that were not committed as a result are aborted, and an API node where any of these transactions originated handles the operations making up those transactions according to its value for `DefaultOperationRedoProblemAction` (by either queuing the operations to be re-tried, or aborting them).

- `RedoOverCommitLimit`

Version (or later)	NDB 8.0.13
Type or units	seconds
Default	20
Range	1 - 4294967039 (0xFFFFFEFF)
Version (or later)	NDB 8.0.19
Type or units	seconds
Default	20
Range	1 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter sets an upper limit in seconds for trying to write a given redo log to disk before timing out. The number of times the data node tries to flush this redo log, but takes longer than `RedoOverCommitLimit`, is kept and compared with `RedoOverCommitCounter`, and when flushing takes too long more times than the value of that parameter, any transactions that were not committed as a result of the flush timeout are aborted. When this occurs, the API node where any of these transactions originated handles the operations making up those transactions according to its `DefaultOperationRedoProblemAction` setting (it either queues the operations to be re-tried, or aborts them).

Controlling restart attempts. It is possible to exercise finely-grained control over restart attempts by data nodes when they fail to start using the `MaxStartFailRetries` and `StartFailRetryDelay` data node configuration parameters.

`MaxStartFailRetries` limits the total number of retries made before giving up on starting the data node, `StartFailRetryDelay` sets the number of seconds between retry attempts. These parameters are listed here:

- `StartFailRetryDelay`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Use this parameter to set the number of seconds between restart attempts by the data node in the event on failure on startup. The default is 0 (no delay).

Both this parameter and `MaxStartFailRetries` are ignored unless `StopOnError` is equal to 0.

- `MaxStartFailRetries`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	3
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Use this parameter to limit the number restart attempts made by the data node in the event that it fails on startup. The default is 3 attempts.

Both this parameter and `StartFailRetryDelay` are ignored unless `StopOnError` is equal to 0.

NDB index statistics parameters. The parameters in the following list relate to NDB index statistics generation.

- `IndexStatAutoCreate`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0, 1
Version (or later)	NDB 8.0.27
Type or units	integer
Default	1
Range	0, 1
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Enable (set equal to 1) or disable (set equal to 0) automatic statistics collection when indexes are created.

- `IndexStatAutoUpdate`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0, 1
Version (or later)	NDB 8.0.27

Type or units	integer
Default	1
Range	0, 1
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Enable (set equal to 1) or disable (set equal to 0) monitoring of indexes for changes, and trigger automatic statistics updates when these are detected. The degree of change needed to trigger the updates are determined by the settings for the `IndexStatTriggerPct` and `IndexStatTriggerScale` options.

- `IndexStatSaveSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	32768
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be restarted with `--initial`. (NDB 8.0.13)

Maximum space in bytes allowed for the saved statistics of any given index in the `NDB` system tables and in the `mysqld` memory cache.

At least one sample is always produced, regardless of any size limit. This size is scaled by `IndexStatSaveScale`.

The size specified by `IndexStatSaveSize` is scaled by the value of `IndexStatTriggerPct` for a large index, times 0.01. This is further multiplied by the logarithm to the base 2 of the index size. Setting `IndexStatTriggerPct` equal to 0 disables the scaling effect.

- `IndexStatSaveScale`

Version (or later)	NDB 8.0.13
Type or units	percentage
Default	100
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Initial Node Restart:** Requires a

	rolling restart of the cluster; each data node must be restarted with `--initial`. (NDB 8.0.13)

The size specified by `IndexStatSaveSize` is scaled by the value of `IndexStatTriggerPct` for a large index, times 0.01. This is further multiplied by the logarithm to the base 2 of the index size. Setting `IndexStatTriggerPct` equal to 0 disables the scaling effect.

- `IndexStatTriggerPct`

Version (or later)	NDB 8.0.13
Type or units	percentage
Default	100
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be restarted with `--initial`. (NDB 8.0.13)

Percentage change in updates that triggers an index statistics update. The value is scaled by `IndexStatTriggerScale`. You can disable this trigger altogether by setting `IndexStatTriggerPct` to 0.

- `IndexStatTriggerScale`

Version (or later)	NDB 8.0.13
Type or units	percentage
Default	100
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be restarted with

	`--initial`. (NDB 8.0.13)

Scale `IndexStatTriggerPct` by this amount times 0.01 for a large index. A value of 0 disables scaling.

- `IndexStatUpdateDelay`

Version (or later)	NDB 8.0.13
Type or units	seconds
Default	60
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Initial Node Restart:** Requires a rolling restart of the cluster; each data node must be restarted with `--initial`. (NDB 8.0.13)

Minimum delay in seconds between automatic index statistics updates for a given index. Setting this variable to 0 disables any delay. The default is 60 seconds.

Restart types. Information about the restart types used by the parameter descriptions in this section is shown in the following table:

Table 23.16 NDB Cluster restart types

Symbol	Restart Type	Description
N	Node	The parameter can be updated using a rolling restart (see Section 23.6.5, "Performing a Rolling Restart of an NDB Cluster")
S	System	All cluster nodes must be shut down completely, then restarted, to effect a change in this parameter
I	Initial	Data nodes must be restarted using the `--initial` option

23.4.3.7 Defining SQL and Other API Nodes in an NDB Cluster

The `[mysqld]` and `[api]` sections in the `config.ini` file define the behavior of the MySQL servers (SQL nodes) and other applications (API nodes) used to access cluster data. None of the parameters shown is required. If no computer or host name is provided, any host can use this SQL or API node.

Generally speaking, a `[mysqld]` section is used to indicate a MySQL server providing an SQL interface to the cluster, and an `[api]` section is used for applications other than `mysqld` processes accessing cluster data, but the two designations are actually synonymous; you can, for instance, list parameters for a MySQL server acting as an SQL node in an `[api]` section.

Note

For a discussion of MySQL server options for NDB Cluster, see MySQL Server Options for NDB Cluster. For information about MySQL server system variables relating to NDB Cluster, see NDB Cluster System Variables.

- `Id`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	[...]
Range	1 - 255
Restart Type	**Initial System Restart:** Requires a complete shutdown of the cluster, wiping and restoring the cluster file system from a backup, and then restarting the cluster. (NDB 8.0.13)

The `Id` is an integer value used to identify the node in all cluster internal messages. The permitted range of values is 1 to 255 inclusive. This value must be unique for each node in the cluster, regardless of the type of node.

Note

In NDB 8.0, data node IDs must be less than 145. If you plan to deploy a large number of data nodes, it is a good idea to limit the node IDs for API nodes (and management nodes) to values greater than 144. (Previously, the maximum supported value for a data node ID was 48.)

`NodeId` is the preferred parameter name to use when identifying API nodes. (`Id` continues to be supported for backward compatibility, but is now deprecated and generates a warning when used. It is also subject to future removal.)

- `ConnectionMap`

Version (or later)	NDB 8.0.13
Type or units	string
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Specifies which data nodes to connect.

- `NodeId`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	[...]
Range	1 - 255
Restart Type	**Initial System Restart:** Requires a complete shutdown of the cluster, wiping and restoring the cluster file system from a backup, and then restarting the cluster. (NDB 8.0.13)

The `NodeId` is an integer value used to identify the node in all cluster internal messages. The permitted range of values is 1 to 255 inclusive. This value must be unique for each node in the cluster, regardless of the type of node.

Note

In NDB 8.0, data node IDs must be less than 145. If you plan to deploy a large number of data nodes, it is a good idea to limit the node IDs for API nodes (and management nodes) to values greater than 144. (Previously, the maximum supported value for a data node ID was 48.)

`NodeId` is the preferred parameter name to use when identifying management nodes. An alias, `Id`, was used for this purpose in very old versions of NDB Cluster, and continues to be supported for backward compatibility; it is now deprecated and generates a warning when used, and is subject to removal in a future release of NDB Cluster.

- `ExecuteOnComputer`

Version (or later)	NDB 8.0.13
Type or units	name
Default	[...]
Range	...
Deprecated	Yes (in NDB 7.5)
Restart Type	**System Restart:** Requires a complete shutdown and restart of the

	cluster. (NDB 8.0.13)

This refers to the `Id` set for one of the computers (hosts) defined in a `[computer]` section of the configuration file.

Important

This parameter is deprecated, and is subject to removal in a future release. Use the `HostName` parameter instead.

-

  The node ID for this node can be given out only to connections that explicitly request it. A management server that requests “any” node ID cannot use this one. This parameter can be used when running multiple management servers on the same host, and `HostName` is not sufficient for distinguishing among processes. Intended for use in testing.

- `HostName`

Version (or later)	NDB 8.0.13
Type or units	name or IP address
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Specifying this parameter defines the hostname of the computer on which the SQL node (API node) is to reside. To specify a hostname, either this parameter or `ExecuteOnComputer` is required.

If no `HostName` or `ExecuteOnComputer` is specified in a given `[mysql]` or `[api]` section of the `config.ini` file, then an SQL or API node may connect using the corresponding “slot” from any host which can establish a network connection to the management server host machine. *This differs from the default behavior for data nodes, where `localhost` is assumed for `HostName` unless otherwise specified*.

- `LocationDomainId`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 16
Restart Type	**System Restart:** Requires a complete shutdown and restart of the

	cluster. (NDB 8.0.13)

Assigns an SQL or other API node to a specific availability domain (also known as an availability zone) within a cloud. By informing `NDB` which nodes are in which availability domains, performance can be improved in a cloud environment in the following ways:

- If requested data is not found on the same node, reads can be directed to another node in the same availability domain.
- Communication between nodes in different availability domains are guaranteed to use `NDB` transporters' WAN support without any further manual intervention.
- The transporter's group number can be based on which availability domain is used, such that also SQL and other API nodes communicate with local data nodes in the same availability domain whenever possible.
- The arbitrator can be selected from an availability domain in which no data nodes are present, or, if no such availability domain can be found, from a third availability domain.

`LocationDomainId` takes an integer value between 0 and 16 inclusive, with 0 being the default; using 0 is the same as leaving the parameter unset.

- `ArbitrationRank`

Version (or later)	NDB 8.0.13
Type or units	0-2
Default	0
Range	0 - 2
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter defines which nodes can act as arbitrators. Both management nodes and SQL nodes can be arbitrators. A value of 0 means that the given node is never used as an arbitrator, a value of 1 gives the node high priority as an arbitrator, and a value of 2 gives it low priority. A normal configuration uses the management server as arbitrator, setting its `ArbitrationRank` to 1 (the default for management nodes) and those for all SQL nodes to 0 (the default for SQL nodes).

By setting `ArbitrationRank` to 0 on all management and SQL nodes, you can disable arbitration completely. You can also control arbitration by overriding this parameter; to do so, set the `Arbitration` parameter in the `[ndbd default]` section of the `config.ini` global configuration file.

- `ArbitrationDelay`

Version (or later)	NDB 8.0.13
Type or units	milliseconds
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)

Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Setting this parameter to any other value than 0 (the default) means that responses by the arbitrator to arbitration requests are delayed by the stated number of milliseconds. It is usually not necessary to change this value.

- `BatchByteSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	16K
Range	1K - 1M
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

For queries that are translated into full table scans or range scans on indexes, it is important for best performance to fetch records in properly sized batches. It is possible to set the proper size both in terms of number of records (`BatchSize`) and in terms of bytes (`BatchByteSize`). The actual batch size is limited by both parameters.

The speed at which queries are performed can vary by more than 40% depending upon how this parameter is set.

This parameter is measured in bytes. The default value is 16K.

- `BatchSize`

Version (or later)	NDB 8.0.13
Type or units	records
Default	256
Range	1 - 992
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter is measured in number of records and is by default set to 256. The maximum size is 992.

- `ExtraSendBufferMemory`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	0

Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter specifies the amount of transporter send buffer memory to allocate in addition to any that has been set using `TotalSendBufferMemory`, `SendBufferMemory`, or both.

- `HeartbeatThreadPriority`

Version (or later)	NDB 8.0.13
Type or units	string
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Use this parameter to set the scheduling policy and priority of heartbeat threads for management and API nodes. The syntax for setting this parameter is shown here:

```
HeartbeatThreadPriority = policy[, priority]

policy:
  {FIFO | RR}
```

When setting this parameter, you must specify a policy. This is one of `FIFO` (first in, first in) or `RR` (round robin). This followed optionally by the priority (an integer).

- `MaxScanBatchSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	256K
Range	32K - 16M
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The batch size is the size of each batch sent from each data node. Most scans are performed in parallel to protect the MySQL Server from receiving too much data from many nodes in parallel; this parameter sets a limit to the total batch size over all nodes.

The default value of this parameter is set to 256KB. Its maximum size is 16MB.

- `TotalSendBufferMemory`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	0
Range	256K - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter is used to determine the total amount of memory to allocate on this node for shared send buffer memory among all configured transporters.

If this parameter is set, its minimum permitted value is 256KB; 0 indicates that the parameter has not been set. For more detailed information, see Section 23.4.3.14, “Configuring NDB Cluster Send Buffer Parameters”.

- `AutoReconnect`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter is `false` by default. This forces disconnected API nodes (including MySQL Servers acting as SQL nodes) to use a new connection to the cluster rather than attempting to re-use an existing one, as re-use of connections can cause problems when using dynamically-allocated node IDs. (Bug #45921)

Note

This parameter can be overridden using the NDB API. For more information, see Ndb_cluster_connection::set_auto_reconnect(), and Ndb_cluster_connection::get_auto_reconnect().

- `DefaultOperationRedoProblemAction`

Version (or later)	NDB 8.0.13
Type or units	enumeration
Default	QUEUE
Range	ABORT, QUEUE

Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter (along with `RedoOverCommitLimit` and `RedoOverCommitCounter`) controls the data node's handling of operations when too much time is taken flushing redo logs to disk. This occurs when a given redo log flush takes longer than `RedoOverCommitLimit` seconds, more than `RedoOverCommitCounter` times, causing any pending transactions to be aborted.

When this happens, the node can respond in either of two ways, according to the value of `DefaultOperationRedoProblemAction`, listed here:

- `ABORT`: Any pending operations from aborted transactions are also aborted.
- `QUEUE`: Pending operations from transactions that were aborted are queued up to be re-tried. This the default. Pending operations are still aborted when the redo log runs out of space—that is, when **`P_TAIL_PROBLEM`** errors occur.

- `DefaultHashMapSize`

Version (or later)	NDB 8.0.13
Type or units	buckets
Default	3840
Range	0 - 3840
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The size of the table hash maps used by `NDB` is configurable using this parameter. `DefaultHashMapSize` can take any of three possible values (0, 240, 3840). These values and their effects are described in the following table.

Table 23.17 DefaultHashMapSize parameter values

Value	Description / Effect
`0`	Use the lowest value set, if any, for this parameter among all data nodes and API nodes in the cluster; if it is not set on any data or API node, use the default value.
`240`	Old default hash map size
`3840`	Hash map size used by default in NDB 8.0

The original intended use for this parameter was to facilitate upgrades and downgrades to and from older NDB Cluster versions, in which the hash map size differed, due to the fact that this change was not otherwise backward compatible. This is not an issue when upgrading to or downgrading from NDB Cluster 8.0.

- `Wan`

Version (or later)	NDB 8.0.13

Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Use WAN TCP setting as default.

- `ConnectBackoffMaxTime`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

In an NDB Cluster with many unstarted data nodes, the value of this parameter can be raised to circumvent connection attempts to data nodes which have not yet begun to function in the cluster, as well as moderate high traffic to management nodes. As long as the API node is not connected to any new data nodes, the value of the `StartConnectBackoffMaxTime` parameter is applied; otherwise, `ConnectBackoffMaxTime` is used to determine the length of time in milliseconds to wait between connection attempts.

Time elapsed *during* node connection attempts is not taken into account when calculating elapsed time for this parameter. The timeout is applied with approximately 100 ms resolution, starting with a 100 ms delay; for each subsequent attempt, the length of this period is doubled until it reaches `ConnectBackoffMaxTime` milliseconds, up to a maximum of 100000 ms (100s).

Once the API node is connected to a data node and that node reports (in a heartbeat message) that it has connected to other data nodes, connection attempts to those data nodes are no longer affected by this parameter, and are made every 100 ms thereafter until connected. Once a data node has started, it can take up `HeartbeatIntervalDbApi` for the API node to be notified that this has occurred.

- `StartConnectBackoffMaxTime`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

In an NDB Cluster with many unstarted data nodes, the value of this parameter can be raised to circumvent connection attempts to data nodes which have not yet begun to function in the cluster, as well as moderate high traffic to management nodes. As long as the API node is not connected to any new data nodes, the value of the `StartConnectBackoffMaxTime` parameter is applied; otherwise, `ConnectBackoffMaxTime` is used to determine the length of time in milliseconds to wait between connection attempts.

Time elapsed *during* node connection attempts is not taken into account when calculating elapsed time for this parameter. The timeout is applied with approximately 100 ms resolution, starting with a 100 ms delay; for each subsequent attempt, the length of this period is doubled until it reaches `StartConnectBackoffMaxTime` milliseconds, up to a maximum of 100000 ms (100s).

Once the API node is connected to a data node and that node reports (in a heartbeat message) that it has connected to other data nodes, connection attempts to those data nodes are no longer affected by this parameter, and are made every 100 ms thereafter until connected. Once a data node has started, it can take up `HeartbeatIntervalDbApi` for the API node to be notified that this has occurred.

API Node Debugging Parameters. You can use the `ApiVerbose` configuration parameter to enable debugging output from a given API node. This parameter takes an integer value. 0 is the default, and disables such debugging; 1 enables debugging output to the cluster log; 2 adds `DBDICT` debugging output as well. (Bug #20638450) See also DUMP 1229.

You can also obtain information from a MySQL server running as an NDB Cluster SQL node using `SHOW STATUS` in the `mysql` client, as shown here:

```
mysql> SHOW STATUS LIKE 'ndb%';
+-----------------------------+----------------+
| Variable_name               | Value          |
+-----------------------------+----------------+
| Ndb_cluster_node_id         | 5              |
| Ndb_config_from_host        | 198.51.100.112 |
| Ndb_config_from_port        | 1186           |
| Ndb_number_of_storage_nodes | 4              |
+-----------------------------+----------------+
4 rows in set (0.02 sec)
```

For information about the status variables appearing in the output from this statement, see NDB Cluster Status Variables.

Note

To add new SQL or API nodes to the configuration of a running NDB Cluster, it is necessary to perform a rolling restart of all cluster nodes after adding new `[mysqld]` or `[api]` sections to the `config.ini` file (or files, if you are using more than one management server). This must be done before the new SQL or API nodes can connect to the cluster.

It is *not* necessary to perform any restart of the cluster if new SQL or API nodes can employ previously unused API slots in the cluster configuration to connect to the cluster.

Restart types. Information about the restart types used by the parameter descriptions in this section is shown in the following table:

Table 23.18 NDB Cluster restart types

Symbol	Restart Type	Description
N	Node	The parameter can be updated using a rolling restart (see

Symbol	Restart Type	Description
		Section 23.6.5, “Performing a Rolling Restart of an NDB Cluster”)
S	System	All cluster nodes must be shut down completely, then restarted, to effect a change in this parameter
I	Initial	Data nodes must be restarted using the `--initial` option

23.4.3.8 Defining the System

The `[system]` section is used for parameters applying to the cluster as a whole. The `Name` system parameter is used with MySQL Enterprise Monitor; `ConfigGenerationNumber` and `PrimaryMGMNode` are not used in production environments. Except when using NDB Cluster with MySQL Enterprise Monitor, is not necessary to have a `[system]` section in the `config.ini` file.

More information about these parameters can be found in the following list:

- `ConfigGenerationNumber`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Configuration generation number. This parameter is currently unused.

- `Name`

Version (or later)	NDB 8.0.13
Type or units	string
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Set a name for the cluster. This parameter is required for deployments with MySQL Enterprise Monitor; it is otherwise unused.

You can obtain the value of this parameter by checking the `Ndb_system_name` status variable. In NDB API applications, you can also retrieve it using `get_system_name()`.

- `PrimaryMGMNode`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Node ID of the primary management node. This parameter is currently unused.

Restart types. Information about the restart types used by the parameter descriptions in this section is shown in the following table:

Table 23.19 NDB Cluster restart types

Symbol	Restart Type	Description
N	Node	The parameter can be updated using a rolling restart (see Section 23.6.5, “Performing a Rolling Restart of an NDB Cluster”)
S	System	All cluster nodes must be shut down completely, then restarted, to effect a change in this parameter
I	Initial	Data nodes must be restarted using the `--initial` option

23.4.3.9 MySQL Server Options and Variables for NDB Cluster

This section provides information about MySQL server options, server and status variables that are specific to NDB Cluster. For general information on using these, and for other options and variables not specific to NDB Cluster, see Section 5.1, “The MySQL Server”.

For NDB Cluster configuration parameters used in the cluster configuration file (usually named `config.ini`), see Section 23.4, “Configuration of NDB Cluster”.

MySQL Server Options for NDB Cluster

This section provides descriptions of `mysqld` server options relating to NDB Cluster. For information about `mysqld` options not specific to NDB Cluster, and for general information about the use of options with `mysqld`, see Section 5.1.7, “Server Command Options”.

For information about command-line options used with other NDB Cluster processes, see Section 23.5, “NDB Cluster Programs”.

- `--ndbcluster`

Command-Line Format	`--ndbcluster[=value]`
Disabled by	`skip-ndbcluster`

Type	Enumeration
Default Value	`ON`

Valid Values	`OFF`
	
	`FORCE`

The `NDBCLUSTER` storage engine is necessary for using NDB Cluster. If a `mysqld` binary includes support for the `NDBCLUSTER` storage engine, the engine is disabled by default. Use the `--ndbcluster` option to enable it. Use `--skip-ndbcluster` to explicitly disable the engine.

The `--ndbcluster` option is ignored (and the `NDB` storage engine is *not* enabled) if `--initialize` is also used. (It is neither necessary nor desirable to use this option together with `--initialize`.)

- `--ndb-allow-copying-alter-table=[ON|OFF]`

Command-Line Format	`--ndb-allow-copying-alter-table[={OFF|ON}]`
System Variable	`ndb_allow_copying_alter_table`
Scope	Global, Session
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value	`ON`

Let `ALTER TABLE` and other DDL statements use copying operations on `NDB` tables. Set to `OFF` to keep this from happening; doing so may improve performance of critical applications.

- `--ndb-applier-allow-skip-epoch`

Command-Line Format	`--ndb-applier-allow-skip-epoch`
Introduced	8.0.28-ndb-8.0.28
System Variable	`ndb_applier_allow_skip_epoch`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No

Use together with `--slave-skip-errors` to cause `NDB` to ignore skipped epoch transactions. Has no effect when used alone.

- `--ndb-batch-size=#`

Command-Line Format	`--ndb-batch-size`
System Variable	`ndb_batch_size`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`32768`
Minimum Value	`0`
Maximum Value (≥ 8.0.29-ndb-8.0.29)	`2147483648`

Maximum Value	2147483648
Maximum Value	2147483648
Maximum Value (≤ 8.0.28-ndb-8.0.28)	31536000
Unit	bytes

This sets the size in bytes that is used for NDB transaction batches.

- `--ndb-cluster-connection-pool=#`

Command-Line Format	--ndb-cluster-connection-pool
System Variable	ndb_cluster_connection_pool
System Variable	ndb_cluster_connection_pool
Scope	Global
Scope	Global
Dynamic	No
Dynamic	No
SET_VAR Hint Applies	No
SET_VAR Hint Applies	No
Type	Integer
Default Value	1
Minimum Value	1
Maximum Value	63

By setting this option to a value greater than 1 (the default), a `mysqld` process can use multiple connections to the cluster, effectively mimicking several SQL nodes. Each connection requires its own `[api]` or `[mysqld]` section in the cluster configuration (`config.ini`) file, and counts against the maximum number of API connections supported by the cluster.

Suppose that you have 2 cluster host computers, each running an SQL node whose `mysqld` process was started with `--ndb-cluster-connection-pool=4`; this means that the cluster must have 8 API slots available for these connections (instead of 2). All of these connections are set up when the SQL node connects to the cluster, and are allocated to threads in a round-robin fashion.

This option is useful only when running `mysqld` on host machines having multiple CPUs, multiple cores, or both. For best results, the value should be smaller than the total number of cores available on the host machine. Setting it to a value greater than this is likely to degrade performance severely.

Important

Because each SQL node using connection pooling occupies multiple API node slots—each slot having its own node ID in the cluster—you must *not* use a node ID as part of the cluster connection string when starting any `mysqld` process that employs connection pooling.

Setting a node ID in the connection string when using the `--ndb-cluster-connection-pool` option causes node ID allocation errors when the SQL node attempts to connect to the cluster.

- `--ndb-cluster-connection-pool-nodeids=list`

Command-Line Format	--ndb-cluster-connection-pool-nodeids
System Variable	ndb_cluster_connection_pool_nodeids

Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	Set
Default Value	

Specifies a comma-separated list of node IDs for connections to the cluster used by an SQL node. The number of nodes in this list must be the same as the value set for the `--ndb-cluster-connection-pool` option.

- `--ndb-blob-read-batch-bytes=bytes`

Command-Line Format	`--ndb-blob-read-batch-bytes`
System Variable	`ndb_blob_read_batch_bytes`
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	`65536`
Minimum Value	`0`
Maximum Value	`4294967295`

This option can be used to set the size (in bytes) for batching of BLOB data reads in NDB Cluster applications. When this batch size is exceeded by the amount of BLOB data to be read within the current transaction, any pending BLOB read operations are immediately executed.

The maximum value for this option is 4294967295; the default is 65536. Setting it to 0 has the effect of disabling BLOB read batching.

Note

In NDB API applications, you can control BLOB write batching with the `setMaxPendingBlobReadBytes()` and `getMaxPendingBlobReadBytes()` methods.

- `--ndb-blob-write-batch-bytes=bytes`

Command-Line Format	`--ndb-blob-write-batch-bytes`
System Variable	`ndb_blob_write_batch_bytes`
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	`65536`
Minimum Value	`0`
Maximum Value	`4294967295`
Unit	bytes

This option can be used to set the size (in bytes) for batching of BLOB data writes in NDB Cluster applications. When this batch size is exceeded by the amount of BLOB data to be written within the current transaction, any pending BLOB write operations are immediately executed.

The maximum value for this option is 4294967295; the default is 65536. Setting it to 0 has the effect of disabling `BLOB` write batching.

Note

In NDB API applications, you can control `BLOB` write batching with the `setMaxPendingBlobWriteBytes()` and `getMaxPendingBlobWriteBytes()` methods.

- `--ndb-connectstring=connection_string`

Command-Line Format	`--ndb-connectstring`
Type	String

When using the `NDBCLUSTER` storage engine, this option specifies the management server that distributes cluster configuration data. See Section 23.4.3.3, “NDB Cluster Connection Strings”, for syntax.

- `--ndb-default-column-format=[FIXED|DYNAMIC]`

Command-Line Format	`--ndb-default-column-format={FIXED|DYNAMIC}`
System Variable	`ndb_default_column_format`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Enumeration
Default Value	`FIXED`

Valid Values	`FIXED`
	
	`DYNAMIC`

Sets the default `COLUMN_FORMAT` and `ROW_FORMAT` for new tables (see Section 13.1.20, “CREATE TABLE Statement”). The default is `FIXED`.

- `--ndb-deferred-constraints=[0|1]`

Command-Line Format	`--ndb-deferred-constraints`
System Variable	`ndb_deferred_constraints`
Scope	Global, Session
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`0`
Minimum Value	`0`
Maximum Value	`1`

Controls whether or not constraint checks on unique indexes are deferred until commit time, where such checks are supported. `0` is the default.

This option is not normally needed for operation of NDB Cluster or NDB Cluster Replication, and is intended primarily for use in testing.

- `--ndb-schema-dist-timeout=#`

Command-Line Format	`--ndb-schema-dist-timeout=#`
Introduced	8.0.17-ndb-8.0.17
System Variable	`ndb_schema_dist_timeout`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`120`
Minimum Value	`5`
Maximum Value	`1200`
Unit	seconds

Specifies the maximum time in seconds that this `mysqld` waits for a schema operation to complete before marking it as having timed out.

- `--ndb-distribution=[KEYHASH|LINHASH]`

Command-Line Format	`--ndb-distribution={KEYHASH|LINHASH}`
System Variable	`ndb_distribution`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Enumeration
Default Value	`KEYHASH`
Valid Values	`LINHASH` `KEYHASH`

Controls the default distribution method for `NDB` tables. Can be set to either of `KEYHASH` (key hashing) or `LINHASH` (linear hashing). `KEYHASH` is the default.

- `--ndb-log-apply-status`

Command-Line Format	`--ndb-log-apply-status[={OFF|ON}]`
System Variable	`ndb_log_apply_status`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value	`OFF`

Causes a replica `mysqld` to log any updates received from its immediate source to the `mysql.ndb_apply_status` table in its own binary log using its own server ID rather than the server ID of the source. In a circular or chain replication setting, this allows such updates to

propagate to the `mysql.ndb_apply_status` tables of any MySQL servers configured as replicas of the current `mysqld`.

In a chain replication setup, using this option allows downstream (replica) clusters to be aware of their positions relative to all of their upstream contributors (sourcess).

In a circular replication setup, this option causes changes to `ndb_apply_status` tables to complete the entire circuit, eventually propagating back to the originating NDB Cluster. This also allows a cluster acting as a replication source to see when its changes (epochs) have been applied to the other clusters in the circle.

This option has no effect unless the MySQL server is started with the `--ndbcluster` option.

- `--ndb-log-empty-epochs=[ON|OFF]`

Command-Line Format	`--ndb-log-empty-epochs[={OFF|ON}]`
System Variable	`ndb_log_empty_epochs`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value	`OFF`

Causes epochs during which there were no changes to be written to the `ndb_apply_status` and `ndb_binlog_index` tables, even when `log_replica_updates` or `log_slave_updates` is enabled.

By default this option is disabled. Disabling `--ndb-log-empty-epochs` causes epoch transactions with no changes not to be written to the binary log, although a row is still written even for an empty epoch in `ndb_binlog_index`.

Because `--ndb-log-empty-epochs=1` causes the size of the `ndb_binlog_index` table to increase independently of the size of the binary log, users should be prepared to manage the growth of this table, even if they expect the cluster to be idle a large part of the time.

- `--ndb-log-empty-update=[ON|OFF]`

Command-Line Format	`--ndb-log-empty-update[={OFF|ON}]`
System Variable	`ndb_log_empty_update`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value	`OFF`

Causes updates that produced no changes to be written to the `ndb_apply_status` and `ndb_binlog_index` tables, even when `log_replica_updates` or `log_slave_updates` is enabled.

By default this option is disabled (`OFF`). Disabling `--ndb-log-empty-update` causes updates with no changes not to be written to the binary log.

- `--ndb-log-exclusive-reads=[0|1]`

Command-Line Format	`--ndb-log-exclusive-reads[={OFF|ON}]`

System Variable	ndb_log_exclusive_reads
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	0

Starting the server with this option causes primary key reads to be logged with exclusive locks, which allows for NDB Cluster Replication conflict detection and resolution based on read conflicts. You can also enable and disable these locks at runtime by setting the value of the ndb_log_exclusive_reads system variable to 1 or 0, respectively. 0 (disable locking) is the default.

For more information, see Read conflict detection and resolution.

- --ndb-log-fail-terminate

Command-Line Format	--ndb-log-fail-terminate
Introduced	8.0.21-ndb-8.0.21
System Variable	ndb_log_fail_terminate
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	Boolean
Default Value	FALSE

When this option is specified, and complete logging of all found row events is not possible, the mysqld process is terminated.

- --ndb-log-orig

Command-Line Format	--ndb-log-orig[={OFF|ON}]
System Variable	ndb_log_orig
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

Log the originating server ID and epoch in the ndb_binlog_index table.

Note

This makes it possible for a given epoch to have multiple rows in ndb_binlog_index, one for each originating epoch.

For more information, see Section 23.7.4, “NDB Cluster Replication Schema and Tables”.

- --ndb-log-transaction-dependency

Command-Line Format	--ndb-log-transaction-dependency={true|false}

Introduced	8.0.33-ndb-8.0.33
System Variable	`ndb_log_transaction_dependency`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value	`false`

Causes the `NDB` binary logging thread to calculate transaction dependencies for each transaction which it writes to the binary log. The default value is `FALSE`.

This option cannot be set at runtime; the corresponding `ndb_log_transaction_dependency` system variable is read-only.

- `--ndb-log-transaction-id`

Command-Line Format	`--ndb-log-transaction-id[={OFF|ON}]`
System Variable	`ndb_log_transaction_id`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value	`OFF`

Causes a replica `mysqld` to write the NDB transaction ID in each row of the binary log. The default value is `FALSE`.

`--ndb-log-transaction-id` is required to enable NDB Cluster Replication conflict detection and resolution using the `NDB$EPOCH_TRANS()` function (see NDB$EPOCH_TRANS()). For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

The deprecated `log_bin_use_v1_row_events` system variable, which defaults to `OFF`, must not be set to `ON` when you use `--ndb-log-transaction-id=ON`.

- `--ndb-log-update-as-write`

Command-Line Format	`--ndb-log-update-as-write[={OFF|ON}]`
System Variable	`ndb_log_update_as_write`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean

Default Value	ON

Whether updates on the source are written to the binary log as updates (OFF) or writes (ON). When this option is enabled, and both --ndb-log-update-as-write and --ndb-log-update-minimal are disabled, operations of different types are logged as described in the following list:

- INSERT: Logged as a WRITE_ROW event with no before image; the after image is logged with all columns.

  UPDATE: Logged as a WRITE_ROW event with no before image; the after image is logged with all columns.

  DELETE: Logged as a DELETE_ROW event with all columns logged in the before image; the after image is not logged.

This option can be used for NDB Replication conflict resolution in combination with the other two NDB logging options mentioned previously; see ndb_replication Table, for more information.

- --ndb-log-updated-only

Command-Line Format	--ndb-log-updated-only[={OFF|ON}]
System Variable	ndb_log_updated_only
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	ON

Whether mysqld writes complete rows (ON) or updates only (OFF) to the binary log. When this option is enabled, and both --ndb-log-update-as-write and --ndb-log-update-minimal are disabled, operations of different types are logged as described in the following list:

- INSERT: Logged as a WRITE_ROW event with no before image; the after image is logged with all columns.
- UPDATE: Logged as an UPDATE_ROW event with primary key columns and updated columns present in both the before and after images.
- DELETE: Logged as a DELETE_ROW event with primary key columns incuded in the before image; the after image is not logged.

This option can be used for NDB Replication conflict resolution in combination with the other two NDB logging options mentioned previously; see ndb_replication Table, for more information about how these options interact with one another.

- --ndb-log-update-minimal

Command-Line Format	--ndb-log-update-minimal[={OFF|ON}]
System Variable	ndb_log_update_minimal
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean

Default Value	OFF

Log updates in a minimal fashion, by writing only the primary key values in the before image, and only the changed columns in the after image. This may cause compatibility problems if replicating to storage engines other than NDB. When this option is enabled, and both --ndb-log-updated-only and --ndb-log-update-as-write are disabled, operations of different types are logged as described in the following list:

- INSERT: Logged as a WRITE_ROW event with no before image; the after image is logged with all columns.
- UPDATE: Logged as an UPDATE_ROW event with primary key columns in the before image; all columns *except* primary key columns are logged in the after image.
- DELETE: Logged as a DELETE_ROW event with all columns in the before image; the after image is not logged.

This option can be used for NDB Replication conflict resolution in combination with the other two NDB logging options mentioned previously; see ndb_replication Table, for more information.

- --ndb-mgmd-host=*host*[:*port*]

Command-Line Format	--ndb-mgmd-host=host_name[:port_num]
Type	String
Default Value	localhost:1186

Can be used to set the host and port number of a single management server for the program to connect to. If the program requires node IDs or references to multiple management servers (or both) in its connection information, use the --ndb-connectstring option instead.

- --ndb-nodeid=#

Command-Line Format	--ndb-nodeid=#
Status Variable	Ndb_cluster_node_id
Scope	Global
Dynamic	No
Type	Integer
Default Value	N/A
Minimum Value	1
Maximum Value	255
Maximum Value	63

Set this MySQL server's node ID in an NDB Cluster.

The --ndb-nodeid option overrides any node ID set with --ndb-connectstring, regardless of the order in which the two options are used.

In addition, if --ndb-nodeid is used, then either a matching node ID must be found in a [mysqld] or [api] section of config.ini, or there must be an “open” [mysqld] or [api] section in the

file (that is, a section without a `NodeId` or `Id` parameter specified). This is also true if the node ID is specified as part of the connection string.

Regardless of how the node ID is determined, its is shown as the value of the global status variable `Ndb_cluster_node_id` in the output of `SHOW STATUS`, and as `cluster_node_id` in the `connection` row of the output of `SHOW ENGINE NDBCLUSTER STATUS`.

For more information about node IDs for NDB Cluster SQL nodes, see Section 23.4.3.7, “Defining SQL and Other API Nodes in an NDB Cluster”.

- `--ndbinfo={ON|OFF|FORCE}`

Command-Line Format	`--ndbinfo[=value]` (≥ 8.0.13-ndb-8.0.13)
Introduced	8.0.13-ndb-8.0.13
Type	Enumeration
Default Value	`ON`

Valid Values	`ON`
	`OFF`
	`FORCE`

Enables the plugin for the `ndbinfo` information database. By default this is ON whenever `NDBCLUSTER` is enabled.

- `--ndb-optimization-delay=`*`milliseconds`*

Command-Line Format	`--ndb-optimization-delay=#`
System Variable	`ndb_optimization_delay`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`10`
Minimum Value	`0`
Maximum Value	`100000`
Unit	milliseconds

Set the number of milliseconds to wait between sets of rows by `OPTIMIZE TABLE` statements on `NDB` tables. The default is 10.

- `--ndb-optimized-node-selection`

Command-Line Format	`--ndb-optimized-node-selection`

Enable optimizations for selection of nodes for transactions. Enabled by default; use `--skip-ndb-optimized-node-selection` to disable.

- `--ndb-transid-mysql-connection-map=`*`state`*

Command-Line Format	`--ndb-transid-mysql-connection-map[=state]`
Type	Enumeration
Default Value	`ON`

Valid Values	`ON`
	`OFF`
	`FORCE`

Enables or disables the plugin that handles the `ndb_transid_mysql_connection_map` table in the `INFORMATION_SCHEMA` database. Takes one of the values `ON`, `OFF`, or `FORCE`. `ON` (the default) enables the plugin. `OFF` disables the plugin, which makes `ndb_transid_mysql_connection_map` inaccessible. `FORCE` keeps the MySQL Server from starting if the plugin fails to load and start.

You can see whether the `ndb_transid_mysql_connection_map` table plugin is running by checking the output of `SHOW PLUGINS`.

- `--ndb-wait-connected=`*`seconds`*

Command-Line Format	`--ndb-wait-connected=#`
System Variable	`ndb_wait_connected`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Integer
Default Value (≥ 8.0.27-ndb-8.0.27)	`120`
Default Value (≤ 8.0.26-ndb-8.0.26)	`30`
Default Value	`30`
Minimum Value	`0`
Maximum Value	`31536000`
Unit	seconds

This option sets the period of time that the MySQL server waits for connections to NDB Cluster management and data nodes to be established before accepting MySQL client connections. The time is specified in seconds. The default value is `30`.

- `--ndb-wait-setup=`*`seconds`*

Command-Line Format	`--ndb-wait-setup=#`
System Variable	`ndb_wait_setup`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Integer
Default Value (≥ 8.0.27-ndb-8.0.27)	`120`
Default Value (≤ 8.0.26-ndb-8.0.26)	`30`
Default Value	`30`
Default Value	`15`
Default Value	`15`
Minimum Value	`0`
Maximum Value	`31536000`

Unit	seconds

This variable shows the period of time that the MySQL server waits for the NDB storage engine to complete setup before timing out and treating NDB as unavailable. The time is specified in seconds. The default value is 30.

- --skip-ndbcluster

Command-Line Format	--skip-ndbcluster

Disable the NDBCLUSTER storage engine. This is the default for binaries that were built with NDBCLUSTER storage engine support; the server allocates memory and other resources for this storage engine only if the --ndbcluster option is given explicitly. See Section 23.4.1, “Quick Test Setup of NDB Cluster”, for an example.

NDB Cluster System Variables

This section provides detailed information about MySQL server system variables that are specific to NDB Cluster and the NDB storage engine. For system variables not specific to NDB Cluster, see Section 5.1.8, “Server System Variables”. For general information on using system variables, see Section 5.1.9, “Using System Variables”.

- ndb_autoincrement_prefetch_sz

Command-Line Format	--ndb-autoincrement-prefetch-sz=#
System Variable	ndb_autoincrement_prefetch_sz
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value (≥ 8.0.19-ndb-8.0.19)	512
Default Value (≤ 8.0.18-ndb-8.0.18)	1
Minimum Value	1
Maximum Value	65536

Determines the probability of gaps in an autoincremented column. Set it to 1 to minimize this. Setting it to a high value for optimization makes inserts faster, but decreases the likelihood of consecutive autoincrement numbers being used in a batch of inserts.

This variable affects only the number of AUTO_INCREMENT IDs that are fetched between statements; within a given statement, at least 32 IDs are obtained at a time.

Important

This variable does not affect inserts performed using INSERT ... SELECT.

- ndb_cache_check_time

Command-Line Format	--ndb-cache-check-time=#
Deprecated	Yes
System Variable	ndb_cache_check_time
Scope	Global
Dynamic	Yes

SET_VAR Hint Applies	No
Type	Integer
Default Value	0
Minimum Value	0
Maximum Value	31536000
Unit	milliseconds

The number of milliseconds that elapse between checks of NDB Cluster SQL nodes by the MySQL query cache. Setting this to 0 (the default and minimum value) means that the query cache checks for validation on every query.

The recommended maximum value for this variable is 1000, which means that the check is performed once per second. A larger value means that the check is performed and possibly invalidated due to updates on different SQL nodes less often. It is generally not desirable to set this to a value greater than 2000.

Note

The query cache ndb_cache_check_time are deprecated in MySQL 5.7; the query cache was removed in MySQL 8.0.

- ndb_clear_apply_status

Command-Line Format	--ndb-clear-apply-status[={OFF|ON}]
System Variable	ndb_clear_apply_status
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	ON

By the default, executing RESET SLAVE causes an NDB Cluster replica to purge all rows from its ndb_apply_status table. You can disable this by setting ndb_clear_apply_status=OFF.

- ndb_conflict_role

Command-Line Format	--ndb-conflict-role=value
Introduced	8.0.23-ndb-8.0.23
System Variable	ndb_conflict_role
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Enumeration
Default Value	NONE
Valid Values	NONE PRIMARY SECONDARY

	PASS

Determines the role of this SQL node (and NDB Cluster) in a circular (“active-active”) replication setup. ndb_slave_conflict_role can take any one of the values PRIMARY, SECONDARY, PASS, or NULL (the default). The replica SQL thread must be stopped before you can change ndb_slave_conflict_role. In addition, it is not possible to change directly between PASS and either of PRIMARY or SECONDARY directly; in such cases, you must ensure that the SQL thread is stopped, then execute SET @@GLOBAL.ndb_slave_conflict_role = 'NONE' first.

This variable replaces ndb_slave_conflict_role, which is deprecated as of NDB 8.0.23.

For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- ndb_data_node_neighbour

Command-Line Format	--ndb-data-node-neighbour=#
System Variable	ndb_data_node_neighbour
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	0
Minimum Value	0
Maximum Value	255

Sets the ID of a “nearest” data node—that is, a preferred nonlocal data node is chosen to execute the transaction, rather than one running on the same host as the SQL or API node. This used to ensure that when a fully replicated table is accessed, we access it on this data node, to ensure that the local copy of the table is always used whenever possible. This can also be used for providing hints for transactions.

This can improve data access times in the case of a node that is physically closer than and thus has higher network throughput than others on the same host.

See Section 13.1.20.12, “Setting NDB Comment Options”, for further information.

Note

An equivalent method set_data_node_neighbour() is provided for use in NDB API applications.

- ndb_dbg_check_shares

Command-Line Format	--ndb-dbg-check-shares=#
Introduced	8.0.13-ndb-8.0.13
System Variable	ndb_dbg_check_shares
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	0
Minimum Value	0

Maximum Value	1

When set to 1, check that no shares are lingering. Available in debug builds only.

- ndb_default_column_format

Command-Line Format	--ndb-default-column-format={FIXED|DYNAMIC}
System Variable	ndb_default_column_format
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Enumeration
Default Value	FIXED
Valid Values	FIXED DYNAMIC

Sets the default COLUMN_FORMAT and ROW_FORMAT for new tables (see Section 13.1.20, “CREATE TABLE Statement”). The default is FIXED.

- ndb_deferred_constraints

Command-Line Format	--ndb-deferred-constraints=#
System Variable	ndb_deferred_constraints
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	0
Minimum Value	0
Maximum Value	1

Controls whether or not constraint checks are deferred, where these are supported. 0 is the default.

This variable is not normally needed for operation of NDB Cluster or NDB Cluster Replication, and is intended primarily for use in testing.

- ndb_distribution

Command-Line Format	--ndb-distribution={KEYHASH|LINHASH}
System Variable	ndb_distribution
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Enumeration
Default Value	KEYHASH
Valid Values	LINHASH

	KEYHASH

Controls the default distribution method for NDB tables. Can be set to either of KEYHASH (key hashing) or LINHASH (linear hashing). KEYHASH is the default.

- ndb_eventbuffer_free_percent

Command-Line Format	--ndb-eventbuffer-free-percent=#
System Variable	ndb_eventbuffer_free_percent
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	20
Minimum Value	1
Maximum Value	99

Sets the percentage of the maximum memory allocated to the event buffer (ndb_eventbuffer_max_alloc) that should be available in event buffer after reaching the maximum, before starting to buffer again.

- ndb_eventbuffer_max_alloc

Command-Line Format	--ndb-eventbuffer-max-alloc=#
System Variable	ndb_eventbuffer_max_alloc
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	0
Minimum Value	0
Maximum Value (≥ 8.0.26-ndb-8.0.26)	9223372036854775807
Maximum Value	9223372036854775807
Maximum Value	9223372036854775807
Maximum Value (≤ 8.0.25-ndb-8.0.25)	4294967295

Sets the maximum amount memory (in bytes) that can be allocated for buffering events by the NDB API. 0 means that no limit is imposed, and is the default.

- ndb_extra_logging

Command-Line Format	ndb_extra_logging=#
System Variable	ndb_extra_logging
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	1

Minimum Value	0
Maximum Value	1

This variable enables recording in the MySQL error log of information specific to the NDB storage engine.

When this variable is set to 0, the only information specific to NDB that is written to the MySQL error log relates to transaction handling. If it set to a value greater than 0 but less than 10, NDB table schema and connection events are also logged, as well as whether or not conflict resolution is in use, and other NDB errors and information. If the value is set to 10 or more, information about NDB internals, such as the progress of data distribution among cluster nodes, is also written to the MySQL error log. The default is 1.

- ndb_force_send

Command-Line Format	--ndb-force-send[={OFF|ON}]
System Variable	ndb_force_send
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	ON

Forces sending of buffers to NDB immediately, without waiting for other threads. Defaults to ON.

- ndb_fully_replicated

Command-Line Format	--ndb-fully-replicated[={OFF|ON}]
System Variable	ndb_fully_replicated
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

Determines whether new NDB tables are fully replicated. This setting can be overridden for an individual table using COMMENT="NDB_TABLE=FULLY_REPLICATED=..." in a CREATE TABLE or ALTER TABLE statement; see Section 13.1.20.12, “Setting NDB Comment Options”, for syntax and other information.

- ndb_index_stat_enable

Command-Line Format	--ndb-index-stat-enable[={OFF|ON}]
System Variable	ndb_index_stat_enable
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean

Default Value	ON

Use NDB index statistics in query optimization. The default is ON.

Prior to NDB 8.0.27, starting the server with --ndb-index-stat-enable set to OFF prevented the creation of the index statistics tables. In NDB 8.0.27 and later, these tables are always created when the server starts, regardless of this option's value.

- ndb_index_stat_option

Command-Line Format	--ndb-index-stat-option=value
System Variable	ndb_index_stat_option
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	String
Default Value	loop_checkon=1000ms,loop_idle=1000ms,loop_b update_batch=1,read_batch=4,idle_batch=32,c check_delay=1m,delete_batch=8,clean_delay=0 error_delay=1m,evict_batch=8,evict_delay=1m cache_lowpct=90

This variable is used for providing tuning options for NDB index statistics generation. The list consist of comma-separated name-value pairs of option names and values, and this list must not contain any space characters.

Options not used when setting ndb_index_stat_option are not changed from their default values. For example, you can set ndb_index_stat_option = 'loop_idle=1000ms,cache_limit=32M'.

Time values can be optionally suffixed with h (hours), m (minutes), or s (seconds). Millisecond values can optionally be specified using ms; millisecond values cannot be specified using h, m, or s.) Integer values can be suffixed with K, M, or G.

The names of the options that can be set using this variable are shown in the table that follows. The table also provides brief descriptions of the options, their default values, and (where applicable) their minimum and maximum values.

Table 23.20 ndb_index_stat_option options and values

Name	Description	Default/Units	Minimum/Maximum
loop_enable		1000 ms	0/4G
loop_idle	Time to sleep when idle	1000 ms	0/4G
loop_busy	Time to sleep when more work is waiting	100 ms	0/4G
update_batch		1	0/4G
read_batch		4	1/4G
idle_batch		32	1/4G
check_batch		8	1/4G
check_delay	How often to check for new statistics	10 m	1/4G
delete_batch		8	0/4G
clean_delay		1 m	0/4G

Name	Description	Default/Units	Minimum/Maximum
error_batch		4	1/4G
error_delay		1 m	1/4G
evict_batch		8	1/4G
evict_delay	Clean LRU cache, from read time	1 m	0/4G
cache_limit	Maximum amount of memory in bytes used for cached index statistics by this mysqld; clean up the cache when this is exceeded.	32 M	0/4G
cache_lowpct		90	0/100
zero_total	Setting this to 1 resets all accumulating counters in ndb_index_stat_status to 0. This option value is also reset to 0 when this is done.	0	0/1

- ndb_join_pushdown

System Variable	ndb_join_pushdown
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	ON

This variable controls whether joins on NDB tables are pushed down to the NDB kernel (data nodes). Previously, a join was handled using multiple accesses of NDB by the SQL node; however, when ndb_join_pushdown is enabled, a pushable join is sent in its entirety to the data nodes, where it can be distributed among the data nodes and executed in parallel on multiple copies of the data, with a single, merged result being returned to mysqld. This can reduce greatly the number of round trips between an SQL node and the data nodes required to handle such a join.

By default, ndb_join_pushdown is enabled.

Conditions for NDB pushdown joins. In order for a join to be pushable, it must meet the following conditions:

1. Only columns can be compared, and all columns to be joined must use *exactly* the same data type. This means that (for example) a join on an INT column and a BIGINT column also cannot be pushed down.

   Previously, expressions such as `t1.a = t2.a + constant` could not be pushed down. This restriction is lifted in NDB 8.0. The result of any operations on any column to be compared must yield the same type as the column itself.

   Expressions comparing columns from the same table can also be pushed down. The columns (or the result of any operations on those columns) must be of exactly the same type, including

the same signedness, length, character set and collation, precision, and scale, where these are applicable.

2. Queries referencing `BLOB` or `TEXT` columns are not supported.

3. Explicit locking is not supported; however, the `NDB` storage engine's characteristic implicit row-based locking is enforced.

   This means that a join using `FOR UPDATE` cannot be pushed down.

4. In order for a join to be pushed down, child tables in the join must be accessed using one of the `ref`, `eq_ref`, or `const` access methods, or some combination of these methods.

   Outer joined child tables can only be pushed using `eq_ref`.

   If the root of the pushed join is an `eq_ref` or `const`, only child tables joined by `eq_ref` can be appended. (A table joined by `ref` is likely to become the root of another pushed join.)

   If the query optimizer decides on `Using join cache` for a candidate child table, that table cannot be pushed as a child. However, it may be the root of another set of pushed tables.

5. Joins referencing tables explicitly partitioned by `[LINEAR] HASH`, `LIST`, or `RANGE` currently cannot be pushed down.

You can see whether a given join can be pushed down by checking it with `EXPLAIN`; when the join can be pushed down, you can see references to the `pushed join` in the `Extra` column of the output, as shown in this example:

```
mysql> EXPLAIN
    ->     SELECT e.first_name, e.last_name, t.title, d.dept_name
    ->         FROM employees e
    ->         JOIN dept_emp de ON e.emp_no=de.emp_no
    ->         JOIN departments d ON d.dept_no=de.dept_no
    ->         JOIN titles t ON e.emp_no=t.emp_no\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: d
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 9
        Extra: Parent of 4 pushed join@1
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: de
         type: ref
possible_keys: PRIMARY,emp_no,dept_no
          key: dept_no
      key_len: 4
          ref: employees.d.dept_no
         rows: 5305
        Extra: Child of 'd' in pushed join@1
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: e
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: employees.de.emp_no
         rows: 1
        Extra: Child of 'de' in pushed join@1
```

```
*************************** 4. row ***************************
           id: 1
  select_type: SIMPLE
        table: t
         type: ref
possible_keys: PRIMARY,emp_no
          key: emp_no
      key_len: 4
          ref: employees.de.emp_no
         rows: 19
        Extra: Child of 'e' in pushed join@1
4 rows in set (0.00 sec)
```

Note

If inner joined child tables are joined by `ref`, *and* the result is ordered or grouped by a sorted index, this index cannot provide sorted rows, which forces writing to a sorted tempfile.

Two additional sources of information about pushed join performance are available:

1. The status variables `Ndb_pushed_queries_defined`, `Ndb_pushed_queries_dropped`, `Ndb_pushed_queries_executed`, and `Ndb_pushed_reads`.
2. The counters in the `ndbinfo.counters` table that belong to the `DBSPJ` kernel block.

- `ndb_log_apply_status`

Command-Line Format	`--ndb-log-apply-status[={OFF|ON}]`
System Variable	`ndb_log_apply_status`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value	`OFF`

A read-only variable which shows whether the server was started with the `--ndb-log-apply-status` option.

- `ndb_log_bin`

Command-Line Format	`--ndb-log-bin[={OFF|ON}]`
System Variable	`ndb_log_bin`
Scope	Global, Session
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value (≥ 8.0.16-ndb-8.0.16)	`OFF`
Default Value (≤ 8.0.15-ndb-8.0.15)	`ON`

Causes updates to `NDB` tables to be written to the binary log. The setting for this variable has no effect if binary logging is not already enabled on the server using `log_bin`. In NDB 8.0, `ndb_log_bin` defaults to 0 (FALSE).

- `ndb_log_binlog_index`

Command-Line Format	`--ndb-log-binlog-index[={OFF|ON}]`

System Variable	ndb_log_binlog_index
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	ON

Causes a mapping of epochs to positions in the binary log to be inserted into the ndb_binlog_index table. Setting this variable has no effect if binary logging is not already enabled for the server using log_bin. (In addition, ndb_log_bin must not be disabled.) ndb_log_binlog_index defaults to 1 (ON); normally, there is never any need to change this value in a production environment.

- ndb_log_empty_epochs

Command-Line Format	--ndb-log-empty-epochs[={OFF|ON}]
System Variable	ndb_log_empty_epochs
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

When this variable is set to 0, epoch transactions with no changes are not written to the binary log, although a row is still written even for an empty epoch in ndb_binlog_index.

- ndb_log_empty_update

Command-Line Format	--ndb-log-empty-update[={OFF|ON}]
System Variable	ndb_log_empty_update
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

When this variable is set to ON (1), update transactions with no changes are written to the binary log, even when log_replica_updates or log_slave_updates is enabled.

- ndb_log_exclusive_reads

Command-Line Format	--ndb-log-exclusive-reads[={OFF|ON}]
System Variable	ndb_log_exclusive_reads
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean

Default Value	0

This variable determines whether primary key reads are logged with exclusive locks, which allows for NDB Cluster Replication conflict detection and resolution based on read conflicts. To enable these locks, set the value of ndb_log_exclusive_reads to 1. 0, which disables such locking, is the default.

For more information, see Read conflict detection and resolution.

- ndb_log_orig

Command-Line Format	--ndb-log-orig[={OFF|ON}]
System Variable	ndb_log_orig
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

Shows whether the originating server ID and epoch are logged in the ndb_binlog_index table. Set using the --ndb-log-orig server option.

- ndb_log_transaction_id

System Variable	ndb_log_transaction_id
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

This read-only, Boolean system variable shows whether a replica mysqld writes NDB transaction IDs in the binary log (required to use “active-active” NDB Cluster Replication with NDB $EPOCH_TRANS() conflict detection). To change the setting, use the --ndb-log-transaction-id option.

ndb_log_transaction_id is not supported in mainline MySQL Server 8.0.

For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- ndb_log_transaction_compression

Command-Line Format	--ndb-log-transaction-compression
Introduced	8.0.31-ndb-8.0.31
System Variable	ndb_log_transaction_compression
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean

Default Value	OFF

Whether a replica mysqld writes compressed transactions in the binary log; present only if mysqld was compiled with support for NDB.

You should note that starting the MySQL server with --binlog-transaction-compression forces this variable to be enabled (ON), and that this overrides any setting for --ndb-log-transaction-compression made on the command line or in a my.cnf file, as shown here:

```
$> mysqld_safe --ndbcluster --ndb-connectstring=127.0.0.1 \
  --binlog-transaction-compression=ON --ndb-log-transaction-compression=OFF &
[1] 27667
$> 2022-07-07T12:29:20.459937Z mysqld_safe Logging to '/usr/local/mysql/data/myhost.err'.
2022-07-07T12:29:20.509873Z mysqld_safe Starting mysqld daemon with databases from /usr/local/mysql/d

$> mysql -e 'SHOW VARIABLES LIKE "%transaction_compression%"'
+-----------------------------------------+-------+
| Variable_name                           | Value |
+-----------------------------------------+-------+
| binlog_transaction_compression          | ON    |
| binlog_transaction_compression_level_zstd  | 3     |
| ndb_log_transaction_compression         | ON    |
| ndb_log_transaction_compression_level_zstd | 3     |
+-----------------------------------------+-------+
```

To disable binary log transaction compression for NDB tables only, set the ndb_log_transaction_compression system variable to OFF in a mysql or other client session after starting mysqld.

Setting the binlog_transaction_compression variable after startup has no effect on the value of ndb_log_transaction_compression.

For more information on binary log transaction compression, such as which events are or are not compressed and as well as behavior changes to be aware of when this feature is used, see Section 5.4.4.5, “Binary Log Transaction Compression”.

- ndb_log_transaction_compression_level_zstd

Command-Line Format	--ndb-log-transaction-compression-level-zstd=#
Introduced	8.0.31-ndb-8.0.31
System Variable	ndb_log_transaction_compression_level_zstd
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	3
Minimum Value	1
Maximum Value	22

The ZSTD compression level used for writing compressed transactions to the replica's binary log if enabled by ndb_log_transaction_compression. Not supported if mysqld was not compiled with support for the NDB storage engine.

See Section 5.4.4.5, “Binary Log Transaction Compression”, for more information.

- ndb_metadata_check

Command-Line Format	--ndb-metadata-check[={OFF|ON}]
Introduced	8.0.16-ndb-8.0.16
System Variable	ndb_metadata_check
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	ON

NDB uses a background thread to check for metadata changes each ndb_metadata_check_interval seconds as compared with the MySQL data dictionary. This metadata change detection thread can be disabled by setting ndb_metadata_check to OFF. The thread is enabled by default.

- ndb_metadata_check_interval

Command-Line Format	--ndb-metadata-check-interval=#
Introduced	8.0.16-ndb-8.0.16
System Variable	ndb_metadata_check_interval
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	60
Minimum Value	0
Maximum Value	31536000
Unit	seconds

NDB runs a metadata change detection thread in the background to determine when the NDB dictionary has changed with respect to the MySQL data dictionary. By default,the interval between such checks is 60 seconds; this can be adjusted by setting the value of ndb_metadata_check_interval. To enable or disable the thread, use ndb_metadata_check.

- ndb_metadata_sync

Introduced	8.0.19-ndb-8.0.19
System Variable	ndb_metadata_sync
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	false

Setting this variable causes the change monitor thread to override any values set for ndb_metadata_check or ndb_metadata_check_interval, and to enter a period of continuous change detection. When the thread ascertains that there are no more changes to be detected, it stalls until the binary logging thread has finished synchronization of all detected objects.

`ndb_metadata_sync` is then set to `false`, and the change monitor thread reverts to the behavior determined by the settings for `ndb_metadata_check` and `ndb_metadata_check_interval`.

In NDB 8.0.22 and later, setting this variable to `true` causes the list of excluded objects to be cleared, and setting it to `false` clears the list of objects to be retried.

- `ndb_optimized_node_selection`

Command-Line Format	`--ndb-optimized-node-selection=#`
System Variable	`ndb_optimized_node_selection`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`3`
Minimum Value	`0`
Maximum Value	`3`

There are two forms of optimized node selection, described here:

1. The SQL node uses *promixity* to determine the transaction coordinator; that is, the “closest” data node to the SQL node is chosen as the transaction coordinator. For this purpose, a data node having a shared memory connection with the SQL node is considered to be “closest” to the SQL node; the next closest (in order of decreasing proximity) are: TCP connection to `localhost`, followed by TCP connection from a host other than `localhost`.

2. The SQL thread uses *distribution awareness* to select the data node. That is, the data node housing the cluster partition accessed by the first statement of a given transaction is used as the transaction coordinator for the entire transaction. (This is effective only if the first statement of the transaction accesses no more than one cluster partition.)

This option takes one of the integer values `0`, `1`, `2`, or `3`. `3` is the default. These values affect node selection as follows:

- `0`: Node selection is not optimized. Each data node is employed as the transaction coordinator 8 times before the SQL thread proceeds to the next data node.

- `1`: Proximity to the SQL node is used to determine the transaction coordinator.

- `2`: Distribution awareness is used to select the transaction coordinator. However, if the first statement of the transaction accesses more than one cluster partition, the SQL node reverts to the round-robin behavior seen when this option is set to `0`.

- `3`: If distribution awareness can be employed to determine the transaction coordinator, then it is used; otherwise proximity is used to select the transaction coordinator. (This is the default behavior.)

Proximity is determined as follows:

1. Start with the value set for the `Group` parameter (default 55).

2. For an API node sharing the same host with other API nodes, decrement the value by 1. Assuming the default value for `Group`, the effective value for data nodes on same host as the API node is 54, and for remote data nodes 55.

3. Setting `ndb_data_node_neighbour` further decreases the effective `Group` value by 50, causing this node to be regarded as the nearest node. This is needed only when all data nodes

are on hosts other than that hosts the API node and it is desirable to dedicate one of them to the API node. In normal cases, the default adjustment described previously is sufficient.

Frequent changes in `ndb_data_node_neighbour` are not advisable, since this changes the state of the cluster connection and thus may disrupt the selection algorithm for new transactions from each thread until it stablilizes.

- `ndb_read_backup`

Command-Line Format	`--ndb-read-backup[={OFF|ON}]`
System Variable	`ndb_read_backup`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value (≥ 8.0.19-ndb-8.0.19)	`ON`
Default Value (≤ 8.0.18-ndb-8.0.18)	`OFF`

Enable read from any fragment replica for any `NDB` table subsequently created; doing so greatly improves the table read performance at a relatively small cost to writes.

If the SQL node and the data node use the same host name or IP address, this fact is detected automatically, so that the preference is to send reads to the same host. If these nodes are on the same host but use different IP addresses, you can tell the SQL node to use the correct data node by setting the value of `ndb_data_node_neighbour` on the SQL node to the node ID of the data node.

To enable or disable read from any fragment replica for an individual table, you can set the `NDB_TABLE` option `READ_BACKUP` for the table accordingly, in a `CREATE TABLE` or `ALTER TABLE` statement; see Section 13.1.20.12, “Setting NDB Comment Options”, for more information.

- `ndb_recv_thread_activation_threshold`

Command-Line Format	`--ndb-recv-thread-activation-threshold=#`
System Variable	`ndb_recv_thread_activation_threshold`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`8`
Minimum Value	`0 (MIN_ACTIVATION_THRESHOLD)`
Maximum Value	`16 (MAX_ACTIVATION_THRESHOLD)`

When this number of concurrently active threads is reached, the receive thread takes over polling of the cluster connection.

This variable is global in scope. It can also be set at startup.

- `ndb_recv_thread_cpu_mask`

Command-Line Format	`--ndb-recv-thread-cpu-mask=mask`
System Variable	`ndb_recv_thread_cpu_mask`
Scope	Global

Dynamic	Yes
SET_VAR Hint Applies	No
Type	Bitmap
Default Value	[empty]

CPU mask for locking receiver threads to specific CPUs. This is specified as a hexadecimal bitmask. For example, 0x33 means that one CPU is used per receiver thread. An empty string is the default; setting ndb_recv_thread_cpu_mask to this value removes any receiver thread locks previously set.

This variable is global in scope. It can also be set at startup.

- ndb_report_thresh_binlog_epoch_slip

Command-Line Format	--ndb-report-thresh-binlog-epoch-slip=#
System Variable	ndb_report_thresh_binlog_epoch_slip
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	10
Minimum Value	0
Maximum Value	256

This represents the threshold for the number of epochs completely buffered in the event buffer, but not yet consumed by the binlog injector thread. When this degree of slippage (lag) is exceeded, an event buffer status message is reported, with BUFFERED_EPOCHS_OVER_THRESHOLD supplied as the reason (see Section 23.6.2.3, “Event Buffer Reporting in the Cluster Log”). Slip is increased when an epoch is received from data nodes and buffered completely in the event buffer; it is decreased when an epoch is consumed by the binlog injector thread, it is reduced. Empty epochs are buffered and queued, and so included in this calculation only when this is enabled using the Ndb::setEventBufferQueueEmptyEpoch() method from the NDB API.

- ndb_report_thresh_binlog_mem_usage

Command-Line Format	--ndb-report-thresh-binlog-mem-usage=#
System Variable	ndb_report_thresh_binlog_mem_usage
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	10
Minimum Value	0
Maximum Value	10

This is a threshold on the percentage of free memory remaining before reporting binary log status. For example, a value of 10 (the default) means that if the amount of available memory for receiving binary log data from the data nodes falls below 10%, a status message is sent to the cluster log.

- `ndb_row_checksum`

System Variable	`ndb_row_checksum`
Scope	Global, Session
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`1`
Minimum Value	`0`
Maximum Value	`1`

Traditionally, `NDB` has created tables with row checksums, which checks for hardware issues at the expense of performance. Setting `ndb_row_checksum` to 0 means that row checksums are *not* used for new or altered tables, which has a significant impact on performance for all types of queries. This variable is set to 1 by default, to provide backward-compatible behavior.

- `ndb_schema_dist_lock_wait_timeout`

Command-Line Format	`--ndb-schema-dist-lock-wait-timeout=value`
Introduced	8.0.18-ndb-8.0.18
System Variable	`ndb_schema_dist_lock_wait_timeout`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`30`
Minimum Value	`0`
Maximum Value	`1200`
Unit	seconds

Number of seconds to wait during schema distribution for the metadata lock taken on each SQL node in order to change its local data dictionary to reflect the DDL statement change. After this time has elapsed, a warning is returned to the effect that a given SQL node's data dictionary was not updated with the change. This avoids having the binary logging thread wait an excessive length of time while handling schema operations.

- `ndb_schema_dist_timeout`

Command-Line Format	`--ndb-schema-dist-timeout=value`
Introduced	8.0.16-ndb-8.0.16
System Variable	`ndb_schema_dist_timeout`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	Integer
Default Value	`120`
Minimum Value	`5`

Maximum Value	1200
Unit	seconds

Number of seconds to wait before detecting a timeout during schema distribution. This can indicate that other SQL nodes are experiencing excessive activity, or that they are somehow being prevented from acquiring necessary resources at this time.

- ndb_schema_dist_upgrade_allowed

Command-Line Format	--ndb-schema-dist-upgrade-allowed=value
Introduced	8.0.17-ndb-8.0.17
System Variable	ndb_schema_dist_upgrade_allowed
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	Boolean
Default Value	true

Allow upgrading of the schema distribution table when connecting to NDB. When true (the default), this change is deferred until all SQL nodes have been upgraded to the same version of the NDB Cluster software.

Note

The performance of the schema distribution may be somewhat degraded until the upgrade has been performed.

- ndb_show_foreign_key_mock_tables

Command-Line Format	--ndb-show-foreign-key-mock-tables[={OFF|ON}]
System Variable	ndb_show_foreign_key_mock_tables
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

Show the mock tables used by NDB to support foreign_key_checks=0. When this is enabled, extra warnings are shown when creating and dropping the tables. The real (internal) name of the table can be seen in the output of SHOW CREATE TABLE.

- ndb_slave_conflict_role

Command-Line Format	--ndb-slave-conflict-role=value
Deprecated	8.0.23-ndb-8.0.23
System Variable	ndb_slave_conflict_role
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No

Type	Enumeration
Default Value	NONE
Valid Values	NONE PRIMARY SECONDARY PASS

Deprecated in NDB 8.0.23, and subject to removal in a future release. Use ndb_conflict_role instead.

- ndb_table_no_logging

System Variable	ndb_table_no_logging
Scope	Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

When this variable is set to ON or 1, it causes NDB tables not to be checkpointed to disk. More specifically, this setting applies to tables which are created or altered using ENGINE NDB when ndb_table_no_logging is enabled, and continues to apply for the lifetime of the table, even if ndb_table_no_logging is later changed. Suppose that A, B, C, and D are tables that we create (and perhaps also alter), and that we also change the setting for ndb_table_no_logging as shown here:

```
SET @@ndb_table_no_logging = 1;

CREATE TABLE A ... ENGINE NDB;

CREATE TABLE B ... ENGINE MYISAM;
CREATE TABLE C ... ENGINE MYISAM;

ALTER TABLE B ENGINE NDB;

SET @@ndb_table_no_logging = 0;

CREATE TABLE D ... ENGINE NDB;
ALTER TABLE C ENGINE NDB;

SET @@ndb_table_no_logging = 1;
```

After the previous sequence of events, tables A and B are not checkpointed; A was created with ENGINE NDB and B was altered to use NDB, both while ndb_table_no_logging was enabled. However, tables C and D are logged; C was altered to use NDB and D was created using ENGINE NDB, both while ndb_table_no_logging was disabled. Setting ndb_table_no_logging back to 1 or ON does *not* cause table C or D to be checkpointed.

Note

ndb_table_no_logging has no effect on the creation of NDB table schema files; to suppress these, use ndb_table_temporary instead.

- ndb_table_temporary

System Variable	ndb_table_temporary

Scope	Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

When set to ON or 1, this variable causes NDB tables not to be written to disk: This means that no table schema files are created, and that the tables are not logged.

Note

Setting this variable currently has no effect. This is a known issue; see Bug #34036.

- ndb_use_copying_alter_table

System Variable	ndb_use_copying_alter_table
Scope	Global, Session
Dynamic	No
SET_VAR Hint Applies	No

Forces NDB to use copying of tables in the event of problems with online ALTER TABLE operations. The default value is OFF.

- ndb_use_exact_count

System Variable	ndb_use_exact_count
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

Forces NDB to use a count of records during SELECT COUNT(*) query planning to speed up this type of query. The default value is OFF, which allows for faster queries overall.

- ndb_use_transactions

Command-Line Format	--ndb-use-transactions[={OFF|ON}]
System Variable	ndb_use_transactions
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	ON

You can disable NDB transaction support by setting this variable's value to OFF. This is generally not recommended, although it may be useful to disable transaction support within a given client session when that session is used to import one or more dump files with large transactions; this allows a multi-row insert to be executed in parts, rather than as a single transaction. In such cases, once the import has been completed, you should either reset the variable value for this session to ON, or simply terminate the session.

- `ndb_version`

System Variable	`ndb_version`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	String
Default Value	

`NDB` engine version, as a composite integer.

- `ndb_version_string`

System Variable	`ndb_version_string`
Scope	Global
Dynamic	No
`SET_VAR` Hint Applies	No
Type	String
Default Value	

`NDB` engine version in `ndb-x.y.z` format.

- `replica_allow_batching`

Command-Line Format	`--replica-allow-batching[={OFF|ON}]`
Introduced	8.0.26-ndb-8.0.26
System Variable	`replica_allow_batching`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value (≥ 8.0.30-ndb-8.0.30)	`ON`
Default Value (≤ 8.0.29-ndb-8.0.29)	`OFF`

Whether or not batched updates are enabled on NDB Cluster replicas. Beginning with NDB 8.0.26, you should use `replica_allow_batching` in place of `slave_allow_batching`, which is deprecated in that release.

Allowing batched updates on the replica greatly improves performance, particularly when replicating `TEXT`, `BLOB`, and `JSON` columns. For this reason, `replica_allow_batching` is enabled by default in NDB 8.0.30 and later.

Setting this variable has an effect only when using replication with the `NDB` storage engine; in MySQL Server 8.0, it is present but does nothing. For more information, see Section 23.7.6, “Starting NDB Cluster Replication (Single Replication Channel)”.

- `ndb_replica_batch_size`

Command-Line Format	`--ndb-replica-batch-size=#`
Introduced	8.0.30-ndb-8.0.30
System Variable	`ndb_replica_batch_size`

Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	2097152
Minimum Value	0
Maximum Value	2097152
Unit	bytes

Determines the batch size in bytes used for writes applied on the replica. In NDB 8.0.30 and later, set this variable rather than the --ndb-batch-size option to apply this setting to the replica, exclusive of any other sessions.

- ndb_replica_blob_write_batch_bytes

Command-Line Format	--ndb-replica-blob-write-batch-bytes=#
Introduced	8.0.30-ndb-8.0.30
System Variable	ndb_replica_blob_write_batch_bytes
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	2097152
Minimum Value	0
Maximum Value	2097152
Unit	bytes

Control the batch write size used for blob data by the replication applier.

Beginning with NDB 8.0.30, you should set this variable rather than the --ndb-blob-write-batch-bytes option to control the blob batch write size on the replica, exclusive of any other sessions. The reason for this is that, when ndb_replica_blob_write_batch_bytes is not set,the effective blob batch size (that is, the maximum number of pending bytes to write for blob columns) is determined by the maximum of the default value of ndb_replica_blob_write_batch_bytes and the value set for --ndb-blob-write-batch-bytes.

Setting ndb_replica_blob_write_batch_bytes to 0 means that NDB imposes no limit on the size of blob batch writes on the replica.

- server_id_bits

Command-Line Format	--server-id-bits=#
System Variable	server_id_bits
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	Integer
Default Value	32

Minimum Value	7
Maximum Value	32

This variable indicates the number of least significant bits within the 32-bit `server_id` which actually identify the server. Indicating that the server is actually identified by fewer than 32 bits makes it possible for some of the remaining bits to be used for other purposes, such as storing user data generated by applications using the NDB API's Event API within the `AnyValue` of an `OperationOptions` structure (NDB Cluster uses the `AnyValue` to store the server ID).

When extracting the effective server ID from `server_id` for purposes such as detection of replication loops, the server ignores the remaining bits. The `server_id_bits` variable is used to mask out any irrelevant bits of `server_id` in the I/O and SQL threads when deciding whether an event should be ignored based on the server ID.

This data can be read from the binary log by `mysqlbinlog`, provided that it is run with its own `server_id_bits` variable set to 32 (the default).

If the value of `server_id` greater than or equal to 2 to the power of `server_id_bits`; otherwise, `mysqld` refuses to start.

This system variable is supported only by NDB Cluster. It is not supported in the standard MySQL 8.0 Server.

- `slave_allow_batching`

Command-Line Format	`--slave-allow-batching[={OFF|ON}]`
Deprecated	8.0.26-ndb-8.0.26
System Variable	`slave_allow_batching`
Scope	Global
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean
Default Value (≥ 8.0.30-ndb-8.0.30)	`ON`
Default Value (≤ 8.0.29-ndb-8.0.29)	`OFF`

Whether or not batched updates are enabled on NDB Cluster replicas. Beginning with NDB 8.0.26, this variable is deprecated, and you should use `replica_allow_batching` instead.

Allowing batched updates on the replica greatly improves performance, particularly when replicating `TEXT`, `BLOB`, and `JSON` columns. For this reason, `replica_allow_batching` is `ON` by default in NDB 8.0.30 and later. Also beginning with NDB 8.0.30, a warning is issued whenever this variable is set to `OFF`.

Setting this variable has an effect only when using replication with the `NDB` storage engine; in MySQL Server 8.0, it is present but does nothing. For more information, see Section 23.7.6, “Starting NDB Cluster Replication (Single Replication Channel)”.

- `transaction_allow_batching`

System Variable	`transaction_allow_batching`
Scope	Session
Dynamic	Yes
`SET_VAR` Hint Applies	No
Type	Boolean

Default Value	OFF

When set to 1 or ON, this variable enables batching of statements within the same transaction. To use this variable, autocommit must first be disabled by setting it to 0 or OFF; otherwise, setting transaction_allow_batching has no effect.

It is safe to use this variable with transactions that performs writes only, as having it enabled can lead to reads from the "before" image. You should ensure that any pending transactions are committed (using an explicit COMMIT if desired) before issuing a SELECT.

Important

transaction_allow_batching should not be used whenever there is the possibility that the effects of a given statement depend on the outcome of a previous statement within the same transaction.

This variable is currently supported for NDB Cluster only.

The system variables in the following list all relate to the ndbinfo information database.

- ndbinfo_database

System Variable	ndbinfo_database
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	String
Default Value	ndbinfo

Shows the name used for the NDB information database; the default is ndbinfo. This is a read-only variable whose value is determined at compile time.

- ndbinfo_max_bytes

Command-Line Format	--ndbinfo-max-bytes=#
System Variable	ndbinfo_max_bytes
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Integer
Default Value	0
Minimum Value	0
Maximum Value	65535

Used in testing and debugging only.

- ndbinfo_max_rows

Command-Line Format	--ndbinfo-max-rows=#
System Variable	ndbinfo_max_rows
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No

Type	Integer
Default Value	10
Minimum Value	1
Maximum Value	256

Used in testing and debugging only.

- ndbinfo_offline

System Variable	ndbinfo_offline
Scope	Global
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF

Place the ndbinfo database into offline mode, in which tables and views can be opened even when they do not actually exist, or when they exist but have different definitions in NDB. No rows are returned from such tables (or views).

- ndbinfo_show_hidden

Command-Line Format	--ndbinfo-show-hidden[={OFF|ON}]
System Variable	ndbinfo_show_hidden
Scope	Global, Session
Dynamic	Yes
SET_VAR Hint Applies	No
Type	Boolean
Default Value	OFF
Valid Values	ON OFF

Whether or not the ndbinfo database's underlying internal tables are shown in the mysql client. The default is OFF.

Note

When ndbinfo_show_hidden is enabled, the internal tables are shown in the ndbinfo database only; they are not visible in TABLES or other INFORMATION_SCHEMA tables, regardless of the variable's setting.

- ndbinfo_table_prefix

System Variable	ndbinfo_table_prefix
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	String

Default Value	ndb$

The prefix used in naming the ndbinfo database's base tables (normally hidden, unless exposed by setting ndbinfo_show_hidden). This is a read-only variable whose default value is ndb$; the prefix itself is determined at compile time.

- ndbinfo_version

System Variable	ndbinfo_version
Scope	Global
Dynamic	No
SET_VAR Hint Applies	No
Type	String
Default Value	

Shows the version of the ndbinfo engine in use; read-only.

NDB Cluster Status Variables

This section provides detailed information about MySQL server status variables that relate to NDB Cluster and the NDB storage engine. For status variables not specific to NDB Cluster, and for general information on using status variables, see Section 5.1.10, “Server Status Variables”.

- Handler_discover

  The MySQL server can ask the NDBCLUSTER storage engine if it knows about a table with a given name. This is called discovery. Handler_discover indicates the number of times that tables have been discovered using this mechanism.

- Ndb_api_adaptive_send_deferred_count

  Number of adaptive send calls that were not actually sent.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_adaptive_send_deferred_count_session

  Number of adaptive send calls that were not actually sent.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_adaptive_send_deferred_count_replica

  Number of adaptive send calls that were not actually sent by this replica.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_adaptive_send_deferred_count_slave

  

  Note

  Deprecated in NDB 8.0.23; use Ndb_api_adaptive_send_deferred_count_replica instead.

  Number of adaptive send calls that were not actually sent by this replica.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- Ndb_api_adaptive_send_forced_count

Number of adaptive send calls using forced-send sent by this MySQL Server (SQL node).

For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_adaptive_send_forced_count_session`

  Number of adaptive send calls using forced-send sent in this client session.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_adaptive_send_forced_count_replica`

  Number of adaptive send calls using forced-send sent by this replica.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_adaptive_send_forced_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_adaptive_send_forced_count_replica` instead.

  Number of adaptive send calls using forced-send sent by this replica.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_adaptive_send_unforced_count`

  Number of adaptive send calls without forced-send sent by this MySQL server (SQL node).

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_adaptive_send_unforced_count_session`

  Number of adaptive send calls without forced-send sent in this client session.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_adaptive_send_unforced_count_replica`

  Number of adaptive send calls without forced-send sent by this replica.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_adaptive_send_unforced_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_adaptive_send_unforced_count_replica` instead.

  Number of adaptive send calls without forced-send sent by this replica.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_sent_count_session`

  Amount of data (in bytes) sent to the data nodes in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_sent_count_replica`

  Amount of data (in bytes) sent to the data nodes by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_sent_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_bytes_sent_count_replica` instead.

  Amount of data (in bytes) sent to the data nodes by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_sent_count`

  Amount of data (in bytes) sent to the data nodes by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_received_count_session`

  Amount of data (in bytes) received from the data nodes in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_received_count_replica`

  Amount of data (in bytes) received from the data nodes by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_received_count_slave`

Note

Deprecated in NDB 8.0.23; use `Ndb_api_bytes_received_count_replica` instead.

Amount of data (in bytes) received from the data nodes by this replica.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_bytes_received_count`

  Amount of data (in bytes) received from the data nodes by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_data_count_injector`

  The number of row change events received by the NDB binlog injector thread.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_data_count`

  The number of row change events received by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_nondata_count_injector`

  The number of events received, other than row change events, by the NDB binary log injector thread.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_nondata_count`

  The number of events received, other than row change events, by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_bytes_count_injector`

  The number of bytes of events received by the NDB binlog injector thread.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_event_bytes_count`

  The number of bytes of events received by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pk_op_count_session`

  The number of operations in this client session based on or using primary keys. This includes operations on blob tables, implicit unlock operations, and auto-increment operations, as well as user-visible primary key operations.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pk_op_count_replica`

  The number of operations by this replica based on or using primary keys. This includes operations on blob tables, implicit unlock operations, and auto-increment operations, as well as user-visible primary key operations.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pk_op_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_pk_op_count_replica` instead.

  The number of operations by this replica based on or using primary keys. This includes operations on blob tables, implicit unlock operations, and auto-increment operations, as well as user-visible primary key operations.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pk_op_count`

  The number of operations by this MySQL Server (SQL node) based on or using primary keys. This includes operations on blob tables, implicit unlock operations, and auto-increment operations, as well as user-visible primary key operations.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pruned_scan_count_session`

  The number of scans in this client session that have been pruned to a single partition.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pruned_scan_count_replica`

  The number of scans by this replica that have been pruned to a single partition.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pruned_scan_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_pruned_scan_count_replica` instead.

  The number of scans by this replica that have been pruned to a single partition.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_pruned_scan_count`

  The number of scans by this MySQL Server (SQL node) that have been pruned to a single partition.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_range_scan_count_session`

  The number of range scans that have been started in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_range_scan_count_replica`

  The number of range scans that have been started by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_range_scan_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_range_scan_count_replica` instead.

  The number of range scans that have been started by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_range_scan_count`

  The number of range scans that have been started by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_read_row_count_session`

  The total number of rows that have been read in this client session. This includes all rows read by any primary key, unique key, or scan operation made in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_read_row_count_replica`

  The total number of rows that have been read by this replica. This includes all rows read by any primary key, unique key, or scan operation made by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_read_row_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_read_row_count_replica` instead.

  The total number of rows that have been read by this replica. This includes all rows read by any primary key, unique key, or scan operation made by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_read_row_count`

  The total number of rows that have been read by this MySQL Server (SQL node). This includes all rows read by any primary key, unique key, or scan operation made by this MySQL Server (SQL node).

  You should be aware that this value may not be completely accurate with regard to rows read by `SELECT COUNT(*)` queries, due to the fact that, in this case, the MySQL server actually reads pseudo-rows in the form `[table fragment ID]:[number of rows in fragment]` and sums the rows per fragment for all fragments in the table to derive an estimated count for all rows. `Ndb_api_read_row_count` uses this estimate and not the actual number of rows in the table.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_scan_batch_count_session`

  The number of batches of rows received in this client session. 1 batch is defined as 1 set of scan results from a single fragment.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_scan_batch_count_replica`

  The number of batches of rows received by this replica. 1 batch is defined as 1 set of scan results from a single fragment.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_scan_batch_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_scan_batch_count_replica` instead.

  The number of batches of rows received by this replica. 1 batch is defined as 1 set of scan results from a single fragment.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_scan_batch_count`

  The number of batches of rows received by this MySQL Server (SQL node). 1 batch is defined as 1 set of scan results from a single fragment.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_table_scan_count_session`

  The number of table scans that have been started in this client session, including scans of internal tables,.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_table_scan_count_replica`

  The number of table scans that have been started by this replica, including scans of internal tables.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_table_scan_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_table_scan_count_replica` instead.

  The number of table scans that have been started by this replica, including scans of internal tables.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_table_scan_count`

  The number of table scans that have been started by this MySQL Server (SQL node), including scans of internal tables,.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_abort_count_session`

  The number of transactions aborted in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_abort_count_replica`

  The number of transactions aborted by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_abort_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_trans_abort_count_replica` instead.

  The number of transactions aborted by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_abort_count`

  The number of transactions aborted by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_close_count_session`

  The number of transactions closed in this client session. This value may be greater than the sum of `Ndb_api_trans_commit_count_session` and `Ndb_api_trans_abort_count_session`, since some transactions may have been rolled back.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_close_count_replica`

  The number of transactions closed by this replica. This value may be greater than the sum of `Ndb_api_trans_commit_count_replica` and `Ndb_api_trans_abort_count_replica`, since some transactions may have been rolled back.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_close_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_trans_close_count_replica` instead.

  The number of transactions closed by this replica. This value may be greater than the sum of `Ndb_api_trans_commit_count_replica` and `Ndb_api_trans_abort_count_replica`, since some transactions may have been rolled back.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_close_count`

  The number of transactions closed by this MySQL Server (SQL node). This value may be greater than the sum of `Ndb_api_trans_commit_count` and `Ndb_api_trans_abort_count`, since some transactions may have been rolled back.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_commit_count_session`

  The number of transactions committed in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_commit_count_replica`

  The number of transactions committed by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_commit_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_trans_commit_count_replica` instead.

  The number of transactions committed by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_commit_count`

  The number of transactions committed by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_local_read_row_count_session`

  The total number of rows that have been read in this client session. This includes all rows read by any primary key, unique key, or scan operation made in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_local_read_row_count_replica`

  The total number of rows that have been read by this replica. This includes all rows read by any primary key, unique key, or scan operation made by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_local_read_row_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_trans_local_read_row_count_replica` instead.

  The total number of rows that have been read by this replica. This includes all rows read by any primary key, unique key, or scan operation made by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_local_read_row_count`

  The total number of rows that have been read by this MySQL Server (SQL node). This includes all rows read by any primary key, unique key, or scan operation made by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_start_count_session`

  The number of transactions started in this client session.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_start_count_replica`

  The number of transactions started by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_start_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_trans_start_count_replica` instead.

  The number of transactions started by this replica.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_trans_start_count`

  The number of transactions started by this MySQL Server (SQL node).

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_uk_op_count_session`

  The number of operations in this client session based on or using unique keys.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

- `Ndb_api_uk_op_count_replica`

  The number of operations by this replica based on or using unique keys.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_uk_op_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_uk_op_count_replica` instead.

  The number of operations by this replica based on or using unique keys.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_uk_op_count`

  The number of operations by this MySQL Server (SQL node) based on or using unique keys.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_exec_complete_count_session`

  The number of times a thread has been blocked in this client session while waiting for execution of an operation to complete. This includes all `execute()` calls as well as implicit executes for blob and auto-increment operations not visible to clients.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_exec_complete_count_replica`

  The number of times a thread has been blocked by this replica while waiting for execution of an operation to complete. This includes all `execute()` calls as well as implicit executes for blob and auto-increment operations not visible to clients.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_exec_complete_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_wait_exec_complete_count_replica` instead.

The number of times a thread has been blocked by this replica while waiting for execution of an operation to complete. This includes all `execute()` calls as well as implicit executes for blob and auto-increment operations not visible to clients.

Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_exec_complete_count`

  The number of times a thread has been blocked by this MySQL Server (SQL node) while waiting for execution of an operation to complete. This includes all `execute()` calls as well as implicit executes for blob and auto-increment operations not visible to clients.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_meta_request_count_session`

  The number of times a thread has been blocked in this client session waiting for a metadata-based signal, such as is expected for DDL requests, new epochs, and seizure of transaction records.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_meta_request_count_replica`

  The number of times a thread has been blocked by this replica waiting for a metadata-based signal, such as is expected for DDL requests, new epochs, and seizure of transaction records.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_meta_request_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_wait_meta_request_count_replica` instead.

  The number of times a thread has been blocked by this replica waiting for a metadata-based signal, such as is expected for DDL requests, new epochs, and seizure of transaction records.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_meta_request_count`

  The number of times a thread has been blocked by this MySQL Server (SQL node) waiting for a metadata-based signal, such as is expected for DDL requests, new epochs, and seizure of transaction records.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_nanos_count_session`

  Total time (in nanoseconds) spent in this client session waiting for any type of signal from the data nodes.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_nanos_count_replica`

  Total time (in nanoseconds) spent by this replica waiting for any type of signal from the data nodes.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_nanos_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_wait_nanos_count_replica` instead.

  Total time (in nanoseconds) spent by this replica waiting for any type of signal from the data nodes.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_nanos_count`

  Total time (in nanoseconds) spent by this MySQL Server (SQL node) waiting for any type of signal from the data nodes.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_scan_result_count_session`

  The number of times a thread has been blocked in this client session while waiting for a scan-based signal, such as when waiting for more results from a scan, or when waiting for a scan to close.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it relates to the current session only, and is not affected by any other clients of this `mysqld`.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_scan_result_count_replica`

  The number of times a thread has been blocked by this replica while waiting for a scan-based signal, such as when waiting for more results from a scan, or when waiting for a scan to close.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_scan_result_count_slave`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_api_wait_scan_result_count_replica` instead.

  The number of times a thread has been blocked by this replica while waiting for a scan-based signal, such as when waiting for more results from a scan, or when waiting for a scan to close.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope. If this MySQL server does not act as a replica, or does not use NDB tables, this value is always 0.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_api_wait_scan_result_count`

  The number of times a thread has been blocked by this MySQL Server (SQL node) while waiting for a scan-based signal, such as when waiting for more results from a scan, or when waiting for a scan to close.

  Although this variable can be read using either `SHOW GLOBAL STATUS` or `SHOW SESSION STATUS`, it is effectively global in scope.

  For more information, see Section 23.6.15, “NDB API Statistics Counters and Variables”.

- `Ndb_cluster_node_id`

  If the server is acting as an NDB Cluster node, then the value of this variable its node ID in the cluster.

  If the server is not part of an NDB Cluster, then the value of this variable is 0.

- `Ndb_config_from_host`

  If the server is part of an NDB Cluster, the value of this variable is the host name or IP address of the Cluster management server from which it gets its configuration data.

  If the server is not part of an NDB Cluster, then the value of this variable is an empty string.

- `Ndb_config_from_port`

If the server is part of an NDB Cluster, the value of this variable is the number of the port through which it is connected to the Cluster management server from which it gets its configuration data.

If the server is not part of an NDB Cluster, then the value of this variable is 0.

- `Ndb_config_generation`

  Shows the generation number of the cluster's current configuration. This can be used as an indicator to determine whether the configuration of the cluster has changed since this SQL node last connected to the cluster.

- `Ndb_conflict_fn_epoch`

  Used in NDB Cluster Replication conflict resolution, this variable shows the number of rows found to be in conflict using `NDB$EPOCH()` conflict resolution on a given `mysqld` since the last time it was restarted.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_epoch_trans`

  Used in NDB Cluster Replication conflict resolution, this variable shows the number of rows found to be in conflict using `NDB$EPOCH_TRANS()` conflict resolution on a given `mysqld` since the last time it was restarted.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_epoch2`

  Shows the number of rows found to be in conflict in NDB Cluster Replication conflict resolution, when using `NDB$EPOCH2()`, on the source designated as the primary since the last time it was restarted.

  For more information, see NDB$EPOCH2().

- `Ndb_conflict_fn_epoch2_trans`

  Used in NDB Cluster Replication conflict resolution, this variable shows the number of rows found to be in conflict using `NDB$EPOCH_TRANS2()` conflict resolution on a given `mysqld` since the last time it was restarted.

  For more information, see NDB$EPOCH2_TRANS().

- `Ndb_conflict_fn_max`

  Used in NDB Cluster Replication conflict resolution, this variable shows the number of times that a row was not applied on the current SQL node due to “greatest timestamp wins” conflict resolution since the last time that this `mysqld` was started.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_max_del_win`

  Shows the number of times that a row was rejected on the current SQL node due to NDB Cluster Replication conflict resolution using `NDB$MAX_DELETE_WIN()`, since the last time that this `mysqld` was started.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_max_del_win_ins`

  Shows the number of times that insertion of a row was rejected on the current SQL node due to NDB Cluster Replication conflict resolution using `NDB$MAX_DEL_WIN_INS()`, since the last time that this `mysqld` was started.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_max_ins`

  Used in NDB Cluster Replication conflict resolution, this variable shows the number of times that a row was not inserted on the current SQL node due to “greatest timestamp wins” conflict resolution since the last time that this `mysqld` was started.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_fn_old`

  Used in NDB Cluster Replication conflict resolution, this variable shows the number of times that a row was not applied as the result of “same timestamp wins” conflict resolution on a given `mysqld` since the last time it was restarted.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_last_conflict_epoch`

  The most recent epoch in which a conflict was detected on this replica. You can compare this value with `Ndb_replica_max_replicated_epoch`; if `Ndb_replica_max_replicated_epoch` is greater than `Ndb_conflict_last_conflict_epoch`, no conflicts have yet been detected.

  See Section 23.7.12, “NDB Cluster Replication Conflict Resolution”, for more information.

- `Ndb_conflict_reflected_op_discard_count`

  When using NDB Cluster Replication conflict resolution, this is the number of reflected operations that were not applied on the secondary, due to encountering an error during execution.

  See Section 23.7.12, “NDB Cluster Replication Conflict Resolution”, for more information.

- `Ndb_conflict_reflected_op_prepare_count`

  When using conflict resolution with NDB Cluster Replication, this status variable contains the number of reflected operations that have been defined (that is, prepared for execution on the secondary).

  See Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_refresh_op_count`

  When using conflict resolution with NDB Cluster Replication, this gives the number of refresh operations that have been prepared for execution on the secondary.

  See Section 23.7.12, “NDB Cluster Replication Conflict Resolution”, for more information.

- `Ndb_conflict_last_stable_epoch`

  Number of rows found to be in conflict by a transactional conflict function

  See Section 23.7.12, “NDB Cluster Replication Conflict Resolution”, for more information.

- `Ndb_conflict_trans_row_conflict_count`

  Used in NDB Cluster Replication conflict resolution, this status variable shows the number of rows found to be directly in-conflict by a transactional conflict function on a given `mysqld` since the last time it was restarted.

  Currently, the only transactional conflict detection function supported by NDB Cluster is NDB$EPOCH_TRANS(), so this status variable is effectively the same as `Ndb_conflict_fn_epoch_trans`.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_trans_row_reject_count`

  Used in NDB Cluster Replication conflict resolution, this status variable shows the total number of rows realigned due to being determined as conflicting by a transactional conflict detection function. This includes not only `Ndb_conflict_trans_row_conflict_count`, but any rows in or dependent on conflicting transactions.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_trans_reject_count`

  Used in NDB Cluster Replication conflict resolution, this status variable shows the number of transactions found to be in conflict by a transactional conflict detection function.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_trans_detect_iter_count`

  Used in NDB Cluster Replication conflict resolution, this shows the number of internal iterations required to commit an epoch transaction. Should be (slightly) greater than or equal to `Ndb_conflict_trans_conflict_commit_count`.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_conflict_trans_conflict_commit_count`

  Used in NDB Cluster Replication conflict resolution, this shows the number of epoch transactions committed after they required transactional conflict handling.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_epoch_delete_delete_count`

  When using delete-delete conflict detection, this is the number of delete-delete conflicts detected, where a delete operation is applied, but the indicated row does not exist.

- `Ndb_execute_count`

  Provides the number of round trips to the `NDB` kernel made by operations.

- `Ndb_last_commit_epoch_server`

  The epoch most recently committed by `NDB`.

- `Ndb_last_commit_epoch_session`

  The epoch most recently committed by this `NDB` client.

- `Ndb_metadata_detected_count`

  The number of times since this server was last started that the NDB metadata change detection thread has discovered changes with respect to the MySQL data dictionary.

- `Ndb_metadata_excluded_count`

  The number of metadata objects that the NDB binlog thread has been unable to synchronize on this SQL node since it was last restarted.

  Should an object be excluded, it is not again considered for automatic synchronization until the user corrects the mismatch manually. This can be done by attempting to use the table with a statement such as `SHOW CREATE TABLE table`, `SELECT * FROM table`, or any other statement that would trigger table discovery.

  Prior to NDB 8.0.22, this variable was named `Ndb_metadata_blacklist_size`.

- `Ndb_metadata_synced_count`

  The number of NDB metadata objects which have been synchronized on this SQL node since it was last restarted.

- `Ndb_number_of_data_nodes`

  If the server is part of an NDB Cluster, the value of this variable is the number of data nodes in the cluster.

  If the server is not part of an NDB Cluster, then the value of this variable is 0.

- `Ndb_pushed_queries_defined`

  The total number of joins pushed down to the NDB kernel for distributed handling on the data nodes.

  

  Note

  Joins tested using `EXPLAIN` that can be pushed down contribute to this number.

- `Ndb_pushed_queries_dropped`

  The number of joins that were pushed down to the NDB kernel but that could not be handled there.

- `Ndb_pushed_queries_executed`

  The number of joins successfully pushed down to `NDB` and executed there.

- `Ndb_pushed_reads`

  The number of rows returned to `mysqld` from the NDB kernel by joins that were pushed down.

  

  Note

  Executing `EXPLAIN` on joins that can be pushed down to `NDB` does not add to this number.

- `Ndb_pruned_scan_count`

  This variable holds a count of the number of scans executed by `NDBCLUSTER` since the NDB Cluster was last started where `NDBCLUSTER` was able to use partition pruning.

  Using this variable together with `Ndb_scan_count` can be helpful in schema design to maximize the ability of the server to prune scans to a single table partition, thereby involving replica only a single data node.

- `Ndb_replica_max_replicated_epoch`

  The most recently committed epoch on this replica. You can compare this value with `Ndb_conflict_last_conflict_epoch`; if `Ndb_replica_max_replicated_epoch` is the greater of the two, no conflicts have yet been detected.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_scan_count`

  This variable holds a count of the total number of scans executed by `NDBCLUSTER` since the NDB Cluster was last started.

- `Ndb_slave_max_replicated_epoch`

  

  Note

  Deprecated in NDB 8.0.23; use `Ndb_slave_max_replicated_epoch` instead.

  The most recently committed epoch on this replica. You can compare this value with `Ndb_conflict_last_conflict_epoch`; if `Ndb_slave_max_replicated_epoch` is the greater of the two, no conflicts have yet been detected.

  For more information, see Section 23.7.12, “NDB Cluster Replication Conflict Resolution”.

- `Ndb_system_name`

  If this MySQL Server is connected to an NDB cluster, this read-only variable shows the cluster system name. Otherwise, the value is an empty string.

- `Ndb_trans_hint_count_session`

  The number of transactions using hints that have been started in the current session. Compare with `Ndb_api_trans_start_count_session` to obtain the proportion of all NDB transactions able to use hints.

23.4.3.10 NDB Cluster TCP/IP Connections

TCP/IP is the default transport mechanism for all connections between nodes in an NDB Cluster. Normally it is not necessary to define TCP/IP connections; NDB Cluster automatically sets up such connections for all data nodes, management nodes, and SQL or API nodes.

Note

For an exception to this rule, see Section 23.4.3.11, “NDB Cluster TCP/IP Connections Using Direct Connections”.

To override the default connection parameters, it is necessary to define a connection using one or more `[tcp]` sections in the `config.ini` file. Each `[tcp]` section explicitly defines a TCP/IP connection between two NDB Cluster nodes, and must contain at a minimum the parameters `NodeId1` and `NodeId2`, as well as any connection parameters to override.

It is also possible to change the default values for these parameters by setting them in the `[tcp default]` section.

Important

Any `[tcp]` sections in the `config.ini` file should be listed *last*, following all other sections in the file. However, this is not required for a `[tcp default]` section. This requirement is a known issue with the way in which the `config.ini` file is read by the NDB Cluster management server.

Connection parameters which can be set in `[tcp]` and `[tcp default]` sections of the `config.ini` file are listed here:

- `AllowUnresolvedHostNames`

Version (or later)	NDB 8.0.22
Type or units	boolean
Default	false
Range	true, false
Added	NDB 8.0.22
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

By default, when a management node fails to resolve a host name while trying to connect, this results in a fatal error. This behavior can be overridden by setting `AllowUnresolvedHostNames` to `true` in the `[tcp default]` section of the global configuration file (usually named `config.ini`), in which case failure to resolve a host name is treated as a warning and `ndb_mgmd` startup continues uninterrupted.

- `Checksum`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter is disabled by default. When it is enabled (set to `Y` or `1`), checksums for all messages are calculated before they placed in the send buffer. This feature ensures that messages are not corrupted while waiting in the send buffer, or by the transport mechanism.

- `Group`

When `ndb_optimized_node_selection` is enabled, node proximity is used in some cases to select which node to connect to. This parameter can be used to influence proximity by setting it to a lower value, which is interpreted as “closer”. See the description of the system variable for more information.

- `HostName1`

Version (or later)	NDB 8.0.13
Type or units	name or IP address
Default	[...]

Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The `HostName1` and `HostName2` parameters can be used to specify specific network interfaces to be used for a given TCP connection between two nodes. The values used for these parameters can be host names or IP addresses.

- `HostName2`

Version (or later)	NDB 8.0.13
Type or units	name or IP address
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The `HostName1` and `HostName2` parameters can be used to specify specific network interfaces to be used for a given TCP connection between two nodes. The values used for these parameters can be host names or IP addresses.

- `NodeId1`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	[none]
Range	1 - 255
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

To identify a connection between two nodes it is necessary to provide their node IDs in the `[tcp]` section of the configuration file as the values of `NodeId1` and `NodeId2`. These are the same unique `Id` values for each of these nodes as described in Section 23.4.3.7, “Defining SQL and Other API Nodes in an NDB Cluster”.

- `NodeId2`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	[none]
Range	1 - 255

Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

To identify a connection between two nodes it is necessary to provide their node IDs in the `[tcp]` section of the configuration file as the values of `NodeId1` and `NodeId2`. These are the same unique `Id` values for each of these nodes as described in Section 23.4.3.7, "Defining SQL and Other API Nodes in an NDB Cluster".

- `NodeIdServer`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	[none]
Range	1 - 63
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Set the server side of a TCP connection.

- `OverloadLimit`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

When more than this many unsent bytes are in the send buffer, the connection is considered overloaded.

This parameter can be used to determine the amount of unsent data that must be present in the send buffer before the connection is considered overloaded. See Section 23.4.3.14, "Configuring NDB Cluster Send Buffer Parameters", for more information.

- `PreferIPVersion`

Version (or later)	NDB 8.0.26
Type or units	enumeration
Default	4
Range	4, 6

Added	NDB 8.0.26
Restart Type	**Initial System Restart:** Requires a complete shutdown of the cluster, wiping and restoring the cluster file system from a backup, and then restarting the cluster. (NDB 8.0.13)

Determines the preference of DNS resolution for IP version 4 or version 6. Because the configuration retrieval mechanism employed by NDB Cluster requires that all connections use the same preference, this parameter should be set in the `[tcp default]` of the `config.ini` global configuration file.

- `PreSendChecksum`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

If this parameter and `Checksum` are both enabled, perform pre-send checksum checks, and check all TCP signals between nodes for errors. Has no effect if `Checksum` is not also enabled.

- `Proxy`

Version (or later)	NDB 8.0.13
Type or units	string
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Set a proxy for the TCP connection.

- `ReceiveBufferMemory`

Version (or later)	NDB 8.0.13

Type or units	bytes
Default	2M
Range	16K - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Specifies the size of the buffer used when receiving data from the TCP/IP socket.

The default value of this parameter is 2MB. The minimum possible value is 16KB; the theoretical maximum is 4GB.

- `SendBufferMemory`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	2M
Range	256K - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

TCP transporters use a buffer to store all messages before performing the send call to the operating system. When this buffer reaches 64KB its contents are sent; these are also sent when a round of messages have been executed. To handle temporary overload situations it is also possible to define a bigger send buffer.

If this parameter is set explicitly, then the memory is not dedicated to each transporter; instead, the value used denotes the hard limit for how much memory (out of the total available memory —that is, `TotalSendBufferMemory`) that may be used by a single transporter. For more information about configuring dynamic transporter send buffer memory allocation in NDB Cluster, see Section 23.4.3.14, “Configuring NDB Cluster Send Buffer Parameters”.

The default size of the send buffer is 2MB, which is the size recommended in most situations. The minimum size is 64 KB; the theoretical maximum is 4 GB.

- `SendSignalId`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false (debug builds: true)
Range	true, false

Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

To be able to retrace a distributed message datagram, it is necessary to identify each message. When this parameter is set to `Y`, message IDs are transported over the network. This feature is disabled by default in production builds, and enabled in `-debug` builds.

- `TcpBind_INADDR_ANY`

  Setting this parameter to `TRUE` or `1` binds `IP_ADDR_ANY` so that connections can be made from anywhere (for autogenerated connections). The default is `FALSE` (`0`).

- `TcpSpinTime`

Version (or later)	NDB 8.0.20
Type or units	µsec
Default	0
Range	0 - 2000
Added	NDB 8.0.20
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Controls spin for a TCP transporter; no enable, set to a nonzero value. This works for both the data node and management or SQL node side of the connection.

- `TCP_MAXSEG_SIZE`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	0
Range	0 - 2G
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Determines the size of the memory set during TCP transporter initialization. The default is recommended for most common usage cases.

- `TCP_RCV_BUF_SIZE`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	0

Range	0 - 2G
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Determines the size of the receive buffer set during TCP transporter initialization. The default and minimum value is 0, which allows the operating system or platform to set this value. The default is recommended for most common usage cases.

- `TCP_SND_BUF_SIZE`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	0
Range	0 - 2G
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Determines the size of the send buffer set during TCP transporter initialization. The default and minimum value is 0, which allows the operating system or platform to set this value. The default is recommended for most common usage cases.

Restart types. Information about the restart types used by the parameter descriptions in this section is shown in the following table:

Table 23.21 NDB Cluster restart types

Symbol	Restart Type	Description
N	Node	The parameter can be updated using a rolling restart (see Section 23.6.5, “Performing a Rolling Restart of an NDB Cluster”)
S	System	All cluster nodes must be shut down completely, then restarted, to effect a change in this parameter
I	Initial	Data nodes must be restarted using the `--initial` option

23.4.3.11 NDB Cluster TCP/IP Connections Using Direct Connections

Setting up a cluster using direct connections between data nodes requires specifying explicitly the crossover IP addresses of the data nodes so connected in the `[tcp]` section of the cluster `config.ini` file.

In the following example, we envision a cluster with at least four hosts, one each for a management server, an SQL node, and two data nodes. The cluster as a whole resides on the `172.23.72.*` subnet of a LAN. In addition to the usual network connections, the two data nodes are connected directly using a standard crossover cable, and communicate with one another directly using IP addresses in the `1.1.0.*` address range as shown:

```
# Management Server
[ndb_mgmd]
Id=1
HostName=172.23.72.20

# SQL Node
[mysqld]
Id=2
HostName=172.23.72.21

# Data Nodes
[ndbd]
Id=3
HostName=172.23.72.22

[ndbd]
Id=4
HostName=172.23.72.23

# TCP/IP Connections
[tcp]
NodeId1=3
NodeId2=4
HostName1=1.1.0.1
HostName2=1.1.0.2
```

The `HostName1` and `HostName2` parameters are used only when specifying direct connections.

The use of direct TCP connections between data nodes can improve the cluster's overall efficiency by enabling the data nodes to bypass an Ethernet device such as a switch, hub, or router, thus cutting down on the cluster's latency.

Note

To take the best advantage of direct connections in this fashion with more than two data nodes, you must have a direct connection between each data node and every other data node in the same node group.

23.4.3.12 NDB Cluster Shared-Memory Connections

Communications between NDB cluster nodes are normally handled using TCP/IP. The shared memory (SHM) transporter is distinguished by the fact that signals are transmitted by writing in memory rather than on a socket. The shared-memory transporter (SHM) can improve performance by negating up to 20% of the overhead required by a TCP connection when running an API node (usually an SQL node) and a data node together on the same host. You can enable a shared memory connection in either of the two ways listed here:

- By setting the `UseShm` data node configuration parameter to `1`, and setting `HostName` for the data node and `HostName` for the API node to the same value.
- By using `[shm]` sections in the cluster configuration file, each containing settings for `NodeId1` and `NodeId2`. This method is described in more detail later in this section.

Suppose a cluster is running a data node which has node ID 1 and an SQL node having node ID 51 on the same host computer at 10.0.0.1. To enable an SHM connection between these two nodes, all that is necessary is to insure that the following entries are included in the cluster configuration file:

```
[ndbd]
NodeId=1
HostName=10.0.0.1
UseShm=1

[mysqld]
NodeId=51
HostName=10.0.0.1
```

Important

The two entries just shown are in addition to any other entries and parameter settings needed by the cluster. A more complete example is shown later in this section.

Before starting data nodes that use SHM connections, it is also necessary to make sure that the operating system on each computer hosting such a data node has sufficient memory allocated to shared memory segments. See the documentation for your operating platform for information regarding this. In setups where multiple hosts are each running a data node and an API node, it is possible to enable shared memory on all such hosts by setting `UseShm` in the `[ndbd default]` section of the configuration file. This is shown in the example later in this section.

While not strictly required, tuning for all SHM connections in the cluster can be done by setting one or more of the following parameters in the `[shm default]` section of the cluster configuration (`config.ini`) file:

- `ShmSize`: Shared memory size
- `ShmSpinTime`: Time in µs to spin before sleeping
- `SendBufferMemory`: Size of buffer for signals sent from this node, in bytes.
- `SendSignalId`: Indicates that a signal ID is included in each signal sent through the transporter.
- `Checksum`: Indicates that a checksum is included in each signal sent through the transporter.
- `PreSendChecksum`: Checks of the checksum are made prior to sending the signal; Checksum must also be enabled for this to work

This example shows a simple setup with SHM connections defined on multiple hosts, in an NDB Cluster using 3 computers listed here by host name, hosting the node types shown:

1. `10.0.0.0`: The management server
2. `10.0.0.1`: A data node and an SQL node
3. `10.0.0.2`: A data node and an SQL node

In this scenario, each data node communicates with both the management server and the other data node using TCP transporters; each SQL node uses a shared memory transporter to communicate with the data nodes that is local to it, and a TCP transporter to communicate with the remote data node. A basic configuration reflecting this setup is enabled by the config.ini file whose contents are shown here:

```
[ndbd default]
DataDir=/path/to/datadir
UseShm=1

[shm default]
ShmSize=8M
ShmSpintime=200
SendBufferMemory=4M

[tcp default]
SendBufferMemory=8M

[ndb_mgmd]
NodeId=49
Hostname=10.0.0.0
DataDir=/path/to/datadir

[ndbd]
NodeId=1
Hostname=10.0.0.1
DataDir=/path/to/datadir

[ndbd]
```

```
NodeId=2
Hostname=10.0.0.2
DataDir=/path/to/datadir

[mysqld]
NodeId=51
Hostname=10.0.0.1

[mysqld]
NodeId=52
Hostname=10.0.0.2

[api]
[api]
```

Parameters affecting all shared memory transporters are set in the `[shm default]` section; these can be overridden on a per-connection basis in one or more `[shm]` sections. Each such section must be associated with a given SHM connection using `NodeId1` and `NodeId2`; the values required for these parameters are the node IDs of the two nodes connected by the transporter. You can also identify the nodes by host name using `HostName1` and `HostName2`, but these parameters are not required.

The API nodes for which no host names are set use the TCP transporter to communicate with data nodes independent of the hosts on which they are started; the parameters and values set in the `[tcp default]` section of the configuration file apply to all TCP transporters in the cluster.

For optimum performance, you can define a spin time for the SHM transporter (`ShmSpinTime` parameter); this affects both the data node receiver thread and the poll owner (receive thread or user thread) in `NDB`.

- `Checksum`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	true
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter is a boolean (`Y`/`N`) parameter which is disabled by default. When it is enabled, checksums for all messages are calculated before being placed in the send buffer.

This feature prevents messages from being corrupted while waiting in the send buffer. It also serves as a check against data being corrupted during transport.

- `Group`

Version (or later)	NDB 8.0.13
Type or units	unsigned
Default	35
Range	0 - 200
Restart Type	**Node Restart:** Requires a rolling restart

	of the cluster. (NDB 8.0.13)

Determines the group proximity; a smaller value is interpreted as being closer. The default value is sufficient for most conditions.

- `HostName1`

Version (or later)	NDB 8.0.13
Type or units	name or IP address
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The `HostName1` and `HostName2` parameters can be used to specify specific network interfaces to be used for a given SHM connection between two nodes. The values used for these parameters can be host names or IP addresses.

- `HostName2`

Version (or later)	NDB 8.0.13
Type or units	name or IP address
Default	[...]
Range	...
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

The `HostName1` and `HostName2` parameters can be used to specify specific network interfaces to be used for a given SHM connection between two nodes. The values used for these parameters can be host names or IP addresses.

- `NodeId1`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	[none]
Range	1 - 255
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

To identify a connection between two nodes it is necessary to provide node identifiers for each of them, as `NodeId1` and `NodeId2`.

- `NodeId2`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	[none]
Range	1 - 255
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

To identify a connection between two nodes it is necessary to provide node identifiers for each of them, as `NodeId1` and `NodeId2`.

- `NodeIdServer`

Version (or later)	NDB 8.0.13
Type or units	numeric
Default	[none]
Range	1 - 63
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Identify the server end of a shared memory connection. By default, this is the node ID of the data node.

- `OverloadLimit`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

When more than this many unsent bytes are in the send buffer, the connection is considered overloaded. See Section 23.4.3.14, “Configuring NDB Cluster Send Buffer Parameters”, and Section 23.6.16.64, “The ndbinfo transporters Table”, for more information.

- `PreSendChecksum`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

If this parameter and `Checksum` are both enabled, perform pre-send checksum checks, and check all SHM signals between nodes for errors. Has no effect if `Checksum` is not also enabled.

- `SendBufferMemory`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	2M
Range	256K - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Size (in bytes) of the shared memory buffer for signals sent from this node using a shared memory connection.

- `SendSignalId`

Version (or later)	NDB 8.0.13
Type or units	boolean
Default	false
Range	true, false
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

To retrace the path of a distributed message, it is necessary to provide each message with a unique identifier. Setting this parameter to `Y` causes these message IDs to be transported over the network as well. This feature is disabled by default in production builds, and enabled in `-debug` builds.

- `ShmKey`

Version (or later)	NDB 8.0.13

Type or units	unsigned
Default	0
Range	0 - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

When setting up shared memory segments, a node ID, expressed as an integer, is used to identify uniquely the shared memory segment to use for the communication. There is no default value. If `UseShm` is enabled, the shared memory key is calculated automatically by `NDB`.

- `ShmSize`

Version (or later)	NDB 8.0.13
Type or units	bytes
Default	4M
Range	64K - 4294967039 (0xFFFFFEFF)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

Each SHM connection has a shared memory segment where messages between nodes are placed by the sender and read by the reader. The size of this segment is defined by `ShmSize`. The default value is 4MB.

- `ShmSpinTime`

Version (or later)	NDB 8.0.13
Type or units	integer
Default	0
Range	0 - 2000
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

When receiving, the time to wait before sleeping, in microseconds.

- `SigNum`

Version (or later)	NDB 8.0.13
Type or units	unsigned

Default	[...]
Range	0 - 4294967039 (0xFFFFFEFF)
Deprecated	Yes (in NDB 7.6)
Restart Type	**Node Restart:** Requires a rolling restart of the cluster. (NDB 8.0.13)

This parameter was used formerly to override operating system signal numbers; in NDB 8.0, it is no longer used, and any setting for it is ignored.

Restart types. Information about the restart types used by the parameter descriptions in this section is shown in the following table:

Table 23.22 NDB Cluster restart types

Symbol	Restart Type	Description
N	Node	The parameter can be updated using a rolling restart (see Section 23.6.5, “Performing a Rolling Restart of an NDB Cluster”)
S	System	All cluster nodes must be shut down completely, then restarted, to effect a change in this parameter
I	Initial	Data nodes must be restarted using the `--initial` option

23.4.3.13 Data Node Memory Management

All memory allocation for a data node is performed when the node is started. This ensures that the data node can run in a stable manner without using swap memory, so that `NDB` can be used for latency-sensitive (realtime) applications. The following types of memory are allocated on data node startup:

- Data memory
- Shared global memory
- Redo log buffers
- Job buffers
- Send buffers
- Page cache for disk data records
- Schema transaction memory
- Transaction memory
- Undo log buffer
- Query memory
- Block objects
- Schema memory

- Block data structures
- Long signal memory
- Shared memory communication buffers

The NDB memory manager, which regulates most data node memory, handles the following memory resources:

- Data Memory (DataMemory)
- Redo log buffers (RedoBuffer)
- Job buffers
- Send buffers (SendBufferMemory, TotalSendBufferMemory, ExtraSendBufferMemory)
- Disk Data record page cache (DiskPageBufferMemory, DiskPageBufferEntries)
- Transaction memory (TransactionMemory)
- Query memory
- Disk access records
- File buffers

Each of these resources is set up with a reserved memory area and a maximum memory area. The reserved memory area can be used only by the resource for which it is reserved and cannot be shared with other resources; a given resource can never allocate more than the maximum memory allowed for the resource. A resource that has no maximum memory can expand to use all the shared memory in the memory manager.

The size of the global shared memory for these resources is controlled by the SharedGlobalMemory configuration parameter (default: 128 MB).

Data memory is always reserved and never acquires any memory from shared memory. It is controlled using the DataMemory configuration parameter, whose maximum is 16384 GB. DataMemory is where records are stored, including hash indexes (approximately 15 bytes per row), ordered indexes (10-12 bytes per row per index), and row headers (16-32 bytes per row).

Redo log buffers also use reserved memory only; this is controlled by the RedoBuffer configuration parameter, which sets the size of the redo log buffer per LDM thread. This means that the actual amount of memory used is the value of this parameter multiplied by the number of LDM threads in the data node.

Job buffers use reserved memory only; the size of this memory is calculated by NDB, based on the numbers of threads of various types.

Send buffers have a reserved part but can also allocate an additional 25% of shared global memory. The send buffer reserved size is calculated in two steps:

1. Use the value of the TotalSendBufferMemory configuration parameter (no default value) or the sum of the individual send buffers used by all individual connections to the data node. A data node is connected to all other data nodes, to all API nodes, and to all management nodes. This means that, in a cluster with 2 data nodes, 2 management nodes, and 10 API nodes each data node has 13 node connections. Since the default value for SendBufferMemory for a data node connection is 2 MByte, this works out to 26 MB total.
2. To obtain the total reserved size for the send buffer, the value of the ExtraSendBufferMemory configuration parameter, if any (default value 0). is added to the value obtained in the previous step.

In other words, if TotalSendBufferMemory has been set, the send buffer size is TotalSendBufferMemory + ExtraSendBufferMemory; otherwise, the size of the

send buffer is equal to `([number of node connections] * SendBufferMemory) + ExtraSendBufferMemory`.

The page cache for disk data records uses a reserved resource only; the size of this resource is controlled by the `DiskPageBufferMemory` configuration parameter (default 64 MB). Memory for 32 KB disk page entries is also allocated; the number of these is determined by the `DiskPageBufferEntries` configuration parameter (default 10).

Transaction memory has a reserved part that either is calculated by `NDB`, or is set explicitly using the `TransactionMemory` configuration parameter, introduced in NDB 8.0 (previously, this value was always calculated by `NDB`); transaction memory can also use an unlimited amount of shared global memory. Transaction memory is used for all operational resources handling transactions, scans, locks, scan buffers, and trigger operations. It also holds table rows as they are updated, before the next commit writes them to data memory.

Previously, operational records used dedicated resources whose sizes were controlled by a number of configuration parameters. In NDB 8.0, these are all allocated from a common transaction memory resource and can also use resources from global shared memory. the size of this resource can be controlled using a single `TransactionMemory` configuration parameter.

Reserved memory for undo log buffers can be set using the `InitialLogFileGroup` configuration parameter. If an undo log buffer is created as part of a `CREATE LOGFILE GROUP` SQL statement, the memory is taken from the transaction memory.

A number of resources relating to metadata for Disk Data resources also have no reserved part, and use shared global memory only. Shared global shared memory is thus shared between send buffers, transaction memory, and Disk Data metadata.

If `TransactionMemory` is not set, it is calculated based on the following parameters:

- `MaxNoOfConcurrentOperations`
- `MaxNoOfConcurrentTransactions`
- `MaxNoOfFiredTriggers`
- `MaxNoOfLocalOperations`
- `MaxNoOfConcurrentIndexOperations`
- `MaxNoOfConcurrentScans`
- `MaxNoOfLocalScans`
- `BatchSizePerLocalScan`
- `TransactionBufferMemory`

When `TransactionMemory` is set explicitly, none of the configuration parameters just listed are used to calculate memory size. In addition, the parameters `MaxNoOfConcurrentIndexOperations`, `MaxNoOfFiredTriggers`, `MaxNoOfLocalOperations`, and `MaxNoOfLocalScans` are incompatible with `TransactionMemory` and cannot be set concurrently with it; if `TransactionMemory` is set and any of these four parameters are also set in the `config.ini` configuration file, the management server cannot start. *Note*: Prior to NDB 8.0.29, this restriction was not enforced for `MaxNoOfFiredTriggers`, `MaxNoOfLocalScans`, or `MaxNoOfLocalOperations` (Bug #102509, Bug #32474988).

The `MaxNoOfConcurrentIndexOperations`, `MaxNoOfFiredTriggers`, `MaxNoOfLocalOperations`, and `MaxNoOfLocalScans` parameters are all deprecated in NDB 8.0; you should expect them to be removed from a future release of MySQL NDB Cluster.

Prior to NDB 8.0.29, it was not possible to set any of `MaxNoOfConcurrentTransactions`, `MaxNoOfConcurrentOperations`, or `MaxNoOfConcurrentScans` concurrently with `TransactionMemory`.

The transaction memory resource contains a large number of memory pools. Each memory pool represents an object type and contains a set of objects; each pool includes a reserved part allocated to the pool at startup; this reserved memory is never returned to shared global memory. Reserved records are found using a data structure having only a single level for fast retrieval, which means that a number of records in each pool should be reserved. The number of reserved records in each pool has some impact on performance and reserved memory allocation, but is generally necessary only in certain very advanced use cases to set the reserved sizes explicitly.

The size of the reserved part of the pool can be controlled by setting the following configuration parameters:

- `ReservedConcurrentIndexOperations`
- `ReservedFiredTriggers`
- `ReservedConcurrentOperations`
- `ReservedLocalScans`
- `ReservedConcurrentTransactions`
- `ReservedConcurrentScans`
- `ReservedTransactionBufferMemory`

For any of the parameters just listed that is not set explicitly in `config.ini`, the reserved setting is calculated as 25% of the corresponding maximum setting. For example, if unset, `ReservedConcurrentIndexOperations` is calculated as 25% of `MaxNoOfConcurrentIndexOperations`, and `ReservedLocalScans` is calculated as 25% of `MaxNoOfLocalScans`.

Note

If `ReservedTransactionBufferMemory` is not set, it is calculated as 25% of `TransactionBufferMemory`.

The number of reserved records is per data node; these records are split among the threads handling them (LDM and TC threads) on each node. In most cases, it is sufficient to set `TransactionMemory` alone, and to allow the number of records in pools to be governed by its value.

`MaxNoOfConcurrentScans` limits the number of concurrent scans that can be active in each TC thread. This is important in guarding against cluster overload.

`MaxNoOfConcurrentOperations` limits the number of operations that can be active at any one time in updating transactions. (Simple reads are not affected by this parameter.) This number needs to be limited because it is necessary to preallocate memory for node failure handling, and a resource must be available for handling the maximum number of active operations in one TC thread when contending with node failures. It is imperative that `MaxNoOfConcurrentOperations` be set to the same number on all nodes (this can be done most easily by setting a value for it once, in the `[ndbd default]` section of the `config.ini` global configuration file). While its value can be increased using a rolling restart (see Section 23.6.5, “Performing a Rolling Restart of an NDB Cluster”), decreasing it in this way is not considered safe due to the possibility of a node failure occurring during the rolling restart.

It is possible to limit the size of a single transaction in NDB Cluster through the `MaxDMLOperationsPerTransaction` parameter. If this is not set, the size of one transaction is limited by `MaxNoOfConcurrentOperations` since this parameter limits the total number of concurrent operations per TC thread.

Schema memory size is controlled by the following set of configuration parameters:

- `MaxNoOfSubscriptions`
- `MaxNoOfSubscribers`

- MaxNoOfConcurrentSubOperations
- MaxNoOfAttributes
- MaxNoOfTables
- MaxNoOfOrderedIndexes
- MaxNoOfUniqueHashIndexes
- MaxNoOfTriggers

The number of nodes and the number of LDM threads also have a major impact on the size of schema memory since the number of partitions in each table and each partition (and its fragment replicas) have to be represented in schema memory.

In addition, a number of other records are allocated during startup. These are relatively small. Each block in each thread contains block objects that use memory. This memory size is also normally quite small compared to the other data node memory structures.

23.4.3.14 Configuring NDB Cluster Send Buffer Parameters

The NDB kernel employs a unified send buffer whose memory is allocated dynamically from a pool shared by all transporters. This means that the size of the send buffer can be adjusted as necessary. Configuration of the unified send buffer can accomplished by setting the following parameters:

- **TotalSendBufferMemory.** This parameter can be set for all types of NDB Cluster nodes—that is, it can be set in the [ndbd], [mgm], and [api] (or [mysql]) sections of the config.ini file. It represents the total amount of memory (in bytes) to be allocated by each node for which it is set for use among all configured transporters. If set, its minimum is 256KB; the maximum is 4294967039.

  To be backward-compatible with existing configurations, this parameter takes as its default value the sum of the maximum send buffer sizes of all configured transporters, plus an additional 32KB (one page) per transporter. The maximum depends on the type of transporter, as shown in the following table:

  Table 23.23 Transporter types with maximum send buffer sizes

Transporter	Maximum Send Buffer Size (bytes)
TCP	SendBufferMemory (default = 2M)
SHM	20K

  This enables existing configurations to function in close to the same way as they did with NDB Cluster 6.3 and earlier, with the same amount of memory and send buffer space available to each transporter. However, memory that is unused by one transporter is not available to other transporters.

- **OverloadLimit.** This parameter is used in the config.ini file [tcp] section, and denotes the amount of unsent data (in bytes) that must be present in the send buffer before the connection is considered overloaded. When such an overload condition occurs, transactions that affect the overloaded connection fail with NDB API Error 1218 (Send Buffers overloaded in NDB kernel) until the overload status passes. The default value is 0, in which case the effective overload limit is calculated as SendBufferMemory * 0.8 for a given connection. The maximum value for this parameter is 4G.

- **SendBufferMemory.** This value denotes a hard limit for the amount of memory that may be used by a single transporter out of the entire pool specified by TotalSendBufferMemory. However, the sum of SendBufferMemory for all configured transporters may be greater than the TotalSendBufferMemory that is set for a given node. This is a way to save memory when many nodes are in use, as long as the maximum amount of memory is never required by all transporters at the same time.

You can use the `ndbinfo.transporters` table to monitor send buffer memory usage, and to detect slowdown and overload conditions that can adversely affect performance.

23.4.4 Using High-Speed Interconnects with NDB Cluster

Even before design of `NDBCLUSTER` began in 1996, it was evident that one of the major problems to be encountered in building parallel databases would be communication between the nodes in the network. For this reason, `NDBCLUSTER` was designed from the very beginning to permit the use of a number of different data transport mechanisms. In this Manual, we use the term *transporter* for these.

The NDB Cluster codebase provides for four different transporters:

- *TCP/IP using 100 Mbps or gigabit Ethernet*, as discussed in Section 23.4.3.10, “NDB Cluster TCP/IP Connections”.
- *Direct (machine-to-machine) TCP/IP*; although this transporter uses the same TCP/IP protocol as mentioned in the previous item, it requires setting up the hardware differently and is configured differently as well. For this reason, it is considered a separate transport mechanism for NDB Cluster. See Section 23.4.3.11, “NDB Cluster TCP/IP Connections Using Direct Connections”, for details.
- *Shared memory (SHM)*. For more information about SHM, see Section 23.4.3.12, “NDB Cluster Shared-Memory Connections”.
- *Scalable Coherent Interface (SCI)*.

Note

Using SCI transporters in NDB Cluster requires specialized hardware, software, and MySQL binaries not available with NDB 8.0.

Most users today employ TCP/IP over Ethernet because it is ubiquitous. TCP/IP is also by far the best-tested transporter for use with NDB Cluster.

Regardless of the transporter used, `NDB` attempts to make sure that communication with data node processes is done using chunks that are as large as possible since this benefits all types of data transmission.

23.5 NDB Cluster Programs

Using and managing an NDB Cluster requires several specialized programs, which we describe in this chapter. We discuss the purposes of these programs in an NDB Cluster, how to use the programs, and what startup options are available for each of them.

These programs include the NDB Cluster data, management, and SQL node processes (`ndbd`, `ndbmtd`, `ndb_mgmd`, and `mysqld`) and the management client (`ndb_mgm`).

For information about using `mysqld` as an NDB Cluster process, see Section 23.6.10, “MySQL Server Usage for NDB Cluster”.

Other `NDB` utility, diagnostic, and example programs are included with the NDB Cluster distribution. These include `ndb_restore`, `ndb_show_tables`, and `ndb_config`. These programs are also covered in this section.

23.5.1 ndbd — The NDB Cluster Data Node Daemon

`ndbd` is the process that is used to handle all the data in tables using the NDB Cluster storage engine. This is the process that empowers a data node to accomplish distributed transaction handling, node recovery, checkpointing to disk, online backup, and related tasks.

In an NDB Cluster, a set of `ndbd` processes cooperate in handling data. These processes can execute on the same computer (host) or on different computers. The correspondences between data nodes and Cluster hosts is completely configurable.

Options that can be used with `ndbd` are shown in the following table. Additional descriptions follow the table.

Table 23.24 Command-line options used with the program ndbd

Format	Description	Added, Deprecated, or Removed
`--bind-address=name`	Local bind address	(Supported in all NDB releases based on MySQL 8.0)
`--character-sets-dir=path`	Directory containing character sets	(Supported in all NDB releases based on MySQL 8.0)
`--connect-delay=#`	Obsolete synonym for --connect-retry-delay, which should be used instead of this option	REMOVED: NDB 8.0.28
`--connect-retries=#`	Set the number of times to retry a connection before giving up; 0 means 1 attempt only (and no retries); -1 means continue retrying indefinitely	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retry-delay=#`	Time to wait between attempts to contact a management server, in seconds; 0 means do not wait between attempts	(Supported in all NDB releases based on MySQL 8.0)
`--connect-string=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--core-file`	Write core file on error; used in debugging	(Supported in all NDB releases based on MySQL 8.0)
`--daemon`, `-d`	Start ndbd as daemon (default); override with --nodaemon	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-extra-file=path`	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-file=path`	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-group-suffix=string`	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)
`--filesystem-password=password`	Password for node file system encryption; can be passed from stdin, tty, or my.cnf file	ADDED: 8.0.31
`--filesystem-password-from-stdin={TRUE|FALSE}`	Get password for node file system encryption, passed from stdin	ADDED: 8.0.31
`--foreground`	Run ndbd in foreground, provided for debugging purposes (implies --nodaemon)	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--initial`	Perform initial start of ndbd, including file system cleanup; consult documentation before using this option	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
`--initial-start`	Perform partial initial start (requires --nowait-nodes)	(Supported in all NDB releases based on MySQL 8.0)
`--install[=name]`	Used to install data node process as Windows service; does not apply on other platforms	(Supported in all NDB releases based on MySQL 8.0)
`--logbuffer-size=#`	Control size of log buffer; for use when debugging with many log messages being generated; default is sufficient for normal operations	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-connectstring=connection_string`, `-c connection_string`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-mgmd-host=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--nodaemon`	Do not start ndbd as daemon; provided for testing purposes	(Supported in all NDB releases based on MySQL 8.0)
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--nostart`, `-n`	Do not start ndbd immediately; ndbd waits for command to start from ndb_mgm	(Supported in all NDB releases based on MySQL 8.0)
`--nowait-nodes=list`	Do not wait for these data nodes to start (takes comma-separated list of node IDs); requires --ndb-nodeid	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable	REMOVED: 8.0.31
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)
`--remove[=name]`	Used to remove data node process that was previously installed as Windows service; does not apply on other platforms	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--verbose`, `-v`	Write extra debugging information to node log	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)

Note

All of these options also apply to the multithreaded version of this program (`ndbmtd`) and you may substitute "`ndbmtd`" for "`ndbd`" wherever the latter occurs in this section.

- `--bind-address`

Command-Line Format	`--bind-address=name`
Type	String
Default Value	

Causes `ndbd` to bind to a specific network interface (host name or IP address). This option has no default value.

- `--character-sets-dir`

Command-Line Format	`--character-sets-dir=path`

Directory containing character sets.

- `--connect-delay=#`

Command-Line Format	`--connect-delay=#`
Deprecated	Yes (removed in 8.0.28-ndb-8.0.28)
Type	Numeric
Default Value	`5`
Minimum Value	`0`
Maximum Value	`3600`

Determines the time to wait between attempts to contact a management server when starting (the number of attempts is controlled by the `--connect-retries` option). The default is 5 seconds.

This option is deprecated, and is subject to removal in a future release of NDB Cluster. Use `--connect-retry-delay` instead.

- `--connect-retries=#`

Command-Line Format	`--connect-retries=#`
Type	Numeric
Default Value	`12`
Minimum Value (≥ 8.0.28-ndb-8.0.28)	`-1`

Minimum Value	`-1`
Minimum Value	`-1`
Minimum Value (≤ 8.0.27-ndb-8.0.27)	`0`
Maximum Value	`65535`

Set the number of times to retry a connection before giving up; 0 means 1 attempt only (and no retries). The default is 12 attempts. The time to wait between attempts is controlled by the `--connect-retry-delay` option.

Beginning with NDB 8.0.28, you can set this option to -1, in which case, the data node process continues indefinitely to try to connect.

- `--connect-retry-delay=#`

Command-Line Format	`--connect-retry-delay=#`
Type	Numeric
Default Value	`5`
Minimum Value	`0`
Maximum Value	`4294967295`

Determines the time to wait between attempts to contact a management server when starting (the time between attempts is controlled by the `--connect-retries` option). The default is 5 seconds.

This option takes the place of the `--connect-delay` option, which is now deprecated and subject to removal in a future release of NDB Cluster.

The short form `-r` for this option is deprecated as of NDB 8.0.28, and subject to removal in a future release of NDB Cluster. Use the long form instead.

- `--connect-string`

Command-Line Format	`--connect-string=connection_string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--core-file`

Command-Line Format	`--core-file`

Write core file on error; used in debugging.

- `--daemon`, `-d`

Command-Line Format	`--daemon`

Instructs `ndbd` or `ndbmtd` to execute as a daemon process. This is the default behavior. `--nodaemon` can be used to prevent the process from running as a daemon.

This option has no effect when running `ndbd` or `ndbmtd` on Windows platforms.

- `--defaults-extra-file`

Command-Line Format	`--defaults-extra-file=path`
Type	String

Default Value	[none]

Read given file after global files are read.

- --defaults-file

Command-Line Format	--defaults-file=path
Type	String
Default Value	[none]

Read default options from given file only.

- --defaults-group-suffix

Command-Line Format	--defaults-group-suffix=string
Type	String
Default Value	[none]

Also read groups with concat(group, suffix).

- --filesystem-password

Command-Line Format	--filesystem-password=password
Introduced	8.0.31

Pass the filesystem encryption and decryption password to the data node process using stdin, tty, or the my.cnf file.

Requires EncryptedFileSystem = 1.

For more information, see Section 23.6.14, “File System Encryption for NDB Cluster”.

- --filesystem-password-from-stdin

Command-Line Format	--filesystem-password-from-stdin={TRUE|FALSE}
Introduced	8.0.31

Pass the filesystem encryption and decryption password to the data node process from stdin (only).

Requires EncryptedFileSystem = 1.

For more information, see Section 23.6.14, “File System Encryption for NDB Cluster”.

- --foreground

Command-Line Format	--foreground

Causes ndbd or ndbmtd to execute as a foreground process, primarily for debugging purposes. This option implies the --nodaemon option.

This option has no effect when running ndbd or ndbmtd on Windows platforms.

- `--help`

Command-Line Format	`--help`

Display help text and exit.

- `--initial`

Command-Line Format	`--initial`

Instructs `ndbd` to perform an initial start. An initial start erases any files created for recovery purposes by earlier instances of `ndbd`. It also re-creates recovery log files. On some operating systems, this process can take a substantial amount of time.

An `--initial` start is to be used *only* when starting the `ndbd` process under very special circumstances; this is because this option causes all files to be removed from the NDB Cluster file system and all redo log files to be re-created. These circumstances are listed here:

- When performing a software upgrade which has changed the contents of any files.
- When restarting the node with a new version of `ndbd`.
- As a measure of last resort when for some reason the node restart or system restart repeatedly fails. In this case, be aware that this node can no longer be used to restore data due to the destruction of the data files.

Warning

To avoid the possibility of eventual data loss, it is recommended that you *not* use the `--initial` option together with `StopOnError = 0`. Instead, set `StopOnError` to 0 in `config.ini` only after the cluster has been started, then restart the data nodes normally—that is, without the `--initial` option. See the description of the `StopOnError` parameter for a detailed explanation of this issue. (Bug #24945638)

Use of this option prevents the `StartPartialTimeout` and `StartPartitionedTimeout` configuration parameters from having any effect.

Important

This option does *not* affect backup files that have already been created by the affected node.

Prior to NDB 8.0.21, the `--initial` option also did not affect any Disk Data files. In NDB 8.0.21 and later, when used to perform an initial restart of the cluster, the option causes the removal of all data files associated with Disk Data tablespaces and undo log files associated with log file groups that existed previously on this data node (see Section 23.6.11, “NDB Cluster Disk Data Tables”).

This option also has no effect on recovery of data by a data node that is just starting (or restarting) from data nodes that are already running (unless they also were started with `--initial`, as part of an initial restart). This recovery of data occurs automatically, and requires no user intervention in an NDB Cluster that is running normally.

It is permissible to use this option when starting the cluster for the very first time (that is, before any data node files have been created); however, it is *not* necessary to do so.

- `--initial-start`

Command-Line Format	`--initial-start`

This option is used when performing a partial initial start of the cluster. Each node should be started with this option, as well as `--nowait-nodes`.

Suppose that you have a 4-node cluster whose data nodes have the IDs 2, 3, 4, and 5, and you wish to perform a partial initial start using only nodes 2, 4, and 5—that is, omitting node 3:

```
$> ndbd --ndb-nodeid=2 --nowait-nodes=3 --initial-start
$> ndbd --ndb-nodeid=4 --nowait-nodes=3 --initial-start
$> ndbd --ndb-nodeid=5 --nowait-nodes=3 --initial-start
```

When using this option, you must also specify the node ID for the data node being started with the `--ndb-nodeid` option.

Important

Do not confuse this option with the `--nowait-nodes` option for `ndb_mgmd`, which can be used to enable a cluster configured with multiple management servers to be started without all management servers being online.

- `--install[=name]`

Command-Line Format	`--install[=name]`
Platform Specific	Windows
Type	String
Default Value	`ndbd`

Causes `ndbd` to be installed as a Windows service. Optionally, you can specify a name for the service; if not set, the service name defaults to `ndbd`. Although it is preferable to specify other `ndbd` program options in a `my.ini` or `my.cnf` configuration file, it is possible to use together with `--install`. However, in such cases, the `--install` option must be specified first, before any other options are given, for the Windows service installation to succeed.

It is generally not advisable to use this option together with the `--initial` option, since this causes the data node file system to be wiped and rebuilt every time the service is stopped and started. Extreme care should also be taken if you intend to use any of the other `ndbd` options that affect the starting of data nodes—including `--initial-start`, `--nostart`, and `--nowait-nodes`—together with `--install`, and you should make absolutely certain you fully understand and allow for any possible consequences of doing so.

The `--install` option has no effect on non-Windows platforms.

- `--logbuffer-size=#`

Command-Line Format	`--logbuffer-size=#`
Type	Integer
Default Value	`32768`
Minimum Value	`2048`
Maximum Value	`4294967295`

Sets the size of the data node log buffer. When debugging with high amounts of extra logging, it is possible for the log buffer to run out of space if there are too many log messages, in which case some log messages can be lost. This should not occur during normal operations.

- `--login-path`

Command-Line Format	`--login-path=path`
Type	String
Default Value	`[none]`

Read given path from login file.

- `--ndb-connectstring`

Command-Line Format	`--ndb-connectstring=connection_string`
Type	String
Default Value	`[none]`

Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;][host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf.

- `--ndb-mgmd-host`

Command-Line Format	`--ndb-mgmd-host=connection_string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--ndb-nodeid`

Command-Line Format	`--ndb-nodeid=#`
Type	Integer
Default Value	`[none]`

Set node ID for this node, overriding any ID set by --ndb-connectstring.

- `--ndb-optimized-node-selection`

Command-Line Format	`--ndb-optimized-node-selection`
Removed	8.0.31

Enable optimizations for selection of nodes for transactions. Enabled by default; use `--skip-ndb-optimized-node-selection` to disable.

- `--nodaemon`

Command-Line Format	`--nodaemon`

Prevents `ndbd` or `ndbmtd` from executing as a daemon process. This option overrides the `--daemon` option. This is useful for redirecting output to the screen when debugging the binary.

The default behavior for `ndbd` and `ndbmtd` on Windows is to run in the foreground, making this option unnecessary on Windows platforms, where it has no effect.

- `--no-defaults`

Command-Line Format	`--no-defaults`

Do not read default options from any option file other than login file.

- `--nostart`, `-n`

Command-Line Format	`--nostart`

Instructs `ndbd` not to start automatically. When this option is used, `ndbd` connects to the management server, obtains configuration data from it, and initializes communication objects. However, it does not actually start the execution engine until specifically requested to do so by the management server. This can be accomplished by issuing the proper `START` command in the management client (see Section 23.6.1, “Commands in the NDB Cluster Management Client”).

- `--nowait-nodes=node_id_1[, node_id_2[, ...]]`

Command-Line Format	`--nowait-nodes=list`
Type	String
Default Value	

This option takes a list of data nodes for which the cluster does not wait, prior to starting.

This can be used to start the cluster in a partitioned state. For example, to start the cluster with only half of the data nodes (nodes 2, 3, 4, and 5) running in a 4-node cluster, you can start each `ndbd` process with `--nowait-nodes=3,5`. In this case, the cluster starts as soon as nodes 2 and 4 connect, and does *not* wait `StartPartitionedTimeout` milliseconds for nodes 3 and 5 to connect as it would otherwise.

If you wanted to start up the same cluster as in the previous example without one `ndbd` (say, for example, that the host machine for node 3 has suffered a hardware failure) then start nodes 2, 4, and 5 with `--nowait-nodes=3`. Then the cluster starts as soon as nodes 2, 4, and 5 connect, and does not wait for node 3 to start.

- `--print-defaults`

Command-Line Format	`--print-defaults`

Print program argument list and exit.

- `--remove[=name]`

Command-Line Format	`--remove[=name]`
Platform Specific	Windows
Type	String
Default Value	`ndbd`

Causes an `ndbd` process that was previously installed as a Windows service to be removed. Optionally, you can specify a name for the service to be uninstalled; if not set, the service name defaults to `ndbd`.

The `--remove` option has no effect on non-Windows platforms.

- `--usage`

Command-Line Format	`--usage`

Display help text and exit; same as --help.

- `--verbose`, `-v`

  Causes extra debug output to be written to the node log.

  You can also use `NODELOG DEBUG ON` and `NODELOG DEBUG OFF` to enable and disable this extra logging while the data node is running.

- `--version`

Command-Line Format	`--version`

  Display version information and exit.

`ndbd` generates a set of log files which are placed in the directory specified by `DataDir` in the `config.ini` configuration file.

These log files are listed below. *`node_id`* is and represents the node's unique identifier. For example, `ndb_2_error.log` is the error log generated by the data node whose node ID is `2`.

- `ndb_`*`node_id`*`_error.log` is a file containing records of all crashes which the referenced `ndbd` process has encountered. Each record in this file contains a brief error string and a reference to a trace file for this crash. A typical entry in this file might appear as shown here:

```
Date/Time: Saturday 30 July 2004 - 00:20:01
Type of error: error
Message: Internal program error (failed ndbrequire)
Fault ID: 2341
Problem data: DbtupFixAlloc.cpp
Object of reference: DBTUP (Line: 173)
ProgramName: NDB Kernel
ProcessID: 14909
TraceFile: ndb_2_trace.log.2
***EOM***
```

  Listings of possible `ndbd` exit codes and messages generated when a data node process shuts down prematurely can be found in Data Node Error Messages.

  Important

  The last entry in the error log file is not necessarily the newest one (nor is it likely to be). Entries in the error log are *not* listed in chronological order; rather, they correspond to the order of the trace files as determined in the `ndb_`*`node_id`*`_trace.log.next` file (see below). Error log entries are thus overwritten in a cyclical and not sequential fashion.

- `ndb_`*`node_id`*`_trace.log.`*`trace_id`* is a trace file describing exactly what happened just before the error occurred. This information is useful for analysis by the NDB Cluster development team.

  It is possible to configure the number of these trace files that are created before old files are overwritten. *`trace_id`* is a number which is incremented for each successive trace file.

- `ndb_`*`node_id`*`_trace.log.next` is the file that keeps track of the next trace file number to be assigned.

- `ndb_`*`node_id`*`_out.log` is a file containing any data output by the `ndbd` process. This file is created only if `ndbd` is started as a daemon, which is the default behavior.

- `ndb_`*`node_id`*`.pid` is a file containing the process ID of the `ndbd` process when started as a daemon. It also functions as a lock file to avoid the starting of nodes with the same identifier.

- `ndb_`*`node_id`*`_signal.log` is a file used only in debug versions of `ndbd`, where it is possible to trace all incoming, outgoing, and internal messages with their data in the `ndbd` process.

It is recommended not to use a directory mounted through NFS because in some environments this can cause problems whereby the lock on the `.pid` file remains in effect even after the process has terminated.

To start `ndbd`, it may also be necessary to specify the host name of the management server and the port on which it is listening. Optionally, one may also specify the node ID that the process is to use.

```
$> ndbd --connect-string="nodeid=2;host=ndb_mgmd.mysql.com:1186"
```

See Section 23.4.3.3, “NDB Cluster Connection Strings”, for additional information about this issue. For more information about data node configuration parameters, see Section 23.4.3.6, “Defining NDB Cluster Data Nodes”.

When `ndbd` starts, it actually initiates two processes. The first of these is called the “angel process”; its only job is to discover when the execution process has been completed, and then to restart the `ndbd` process if it is configured to do so. Thus, if you attempt to kill `ndbd` using the Unix `kill` command, it is necessary to kill both processes, beginning with the angel process. The preferred method of terminating an `ndbd` process is to use the management client and stop the process from there.

The execution process uses one thread for reading, writing, and scanning data, as well as all other activities. This thread is implemented asynchronously so that it can easily handle thousands of concurrent actions. In addition, a watch-dog thread supervises the execution thread to make sure that it does not hang in an endless loop. A pool of threads handles file I/O, with each thread able to handle one open file. Threads can also be used for transporter connections by the transporters in the `ndbd` process. In a multi-processor system performing a large number of operations (including updates), the `ndbd` process can consume up to 2 CPUs if permitted to do so.

For a machine with many CPUs it is possible to use several `ndbd` processes which belong to different node groups; however, such a configuration is still considered experimental and is not supported for MySQL 8.0 in a production setting. See Section 23.2.7, “Known Limitations of NDB Cluster”.

23.5.2 ndbinfo_select_all — Select From ndbinfo Tables

`ndbinfo_select_all` is a client program that selects all rows and columns from one or more tables in the `ndbinfo` database

Not all `ndbinfo` tables available in the `mysql` client can be read by this program (see later in this section). In addition, `ndbinfo_select_all` can show information about some tables internal to `ndbinfo` which cannot be accessed using SQL, including the `tables` and `columns` metadata tables.

To select from one or more `ndbinfo` tables using `ndbinfo_select_all`, it is necessary to supply the names of the tables when invoking the program as shown here:

```
$> ndbinfo_select_all table_name1  [table_name2] [...]
```

For example:

```
$> ndbinfo_select_all logbuffers logspaces
== logbuffers ==
node_id log_type        log_id  log_part        total   used    high
5       0       0       0       33554432        262144  0
6       0       0       0       33554432        262144  0
7       0       0       0       33554432        262144  0
8       0       0       0       33554432        262144  0
== logspaces ==
node_id log_type        log_id  log_part        total   used    high
5       0       0       0       268435456       0       0
5       0       0       1       268435456       0       0
5       0       0       2       268435456       0       0
5       0       0       3       268435456       0       0
6       0       0       0       268435456       0       0
6       0       0       1       268435456       0       0
```

```
6       0       0       2       268435456       0       0
6       0       0       3       268435456       0       0
7       0       0       0       268435456       0       0
7       0       0       1       268435456       0       0
7       0       0       2       268435456       0       0
7       0       0       3       268435456       0       0
8       0       0       0       268435456       0       0
8       0       0       1       268435456       0       0
8       0       0       2       268435456       0       0
8       0       0       3       268435456       0       0
$>
```

Options that can be used with `ndbinfo_select_all` are shown in the following table. Additional descriptions follow the table.

Table 23.25 Command-line options used with the program ndbinfo_select_all

Format	Description	Added, Deprecated, or Removed
`--character-sets-dir=path`	Directory containing character sets	REMOVED: 8.0.31
`--connect-retries=#`	Number of times to retry connection before giving up	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retry-delay=#`	Number of seconds to wait between attempts to contact management server	(Supported in all NDB releases based on MySQL 8.0)
`--connect-string=connection-string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--core-file`	Write core file on error; used in debugging	REMOVED: 8.0.31
`--database=db_name`, `-d`	Name of database where table is located	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-extra-file=path`	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-file=path`	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-group-suffix=string`	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)
`--delay=#`	Set delay in seconds between loops	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)
`--loops=#`, `-l`	Set number of times to perform select	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-connectstring=connection-string`, `-c`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
	NDB_CONNECTSTRING and my.cnf	
`--ndb-mgmd-host=connection-string`, `-c`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable	REMOVED: 8.0.31
`--parallelism=#`, `-p`	Set degree of parallelism	(Supported in all NDB releases based on MySQL 8.0)
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)

- `--character-sets-dir`

Command-Line Format	`--character-sets-dir=path`
Removed	8.0.31

Directory containing character sets.

- `--core-file`

Command-Line Format	`--core-file`
Removed	8.0.31

Write core file on error; used in debugging.

- `--connect-retries`

Command-Line Format	`--connect-retries=#`
Type	Integer
Default Value	`12`
Minimum Value	`0`
Maximum Value	`12`

Number of times to retry connection before giving up.

- `--connect-retry-delay`

Command-Line Format	`--connect-retry-delay=#`
Type	Integer
Default Value	`5`
Minimum Value	`0`
Maximum Value	`5`

Number of seconds to wait between attempts to contact management server.

- `--connect-string`

Command-Line Format	`--connect-string=connection-string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--defaults-extra-file`

Command-Line Format	`--defaults-extra-file=path`
Type	String
Default Value	`[none]`

Read given file after global files are read.

- `--defaults-file`

Command-Line Format	`--defaults-file=path`
Type	String
Default Value	`[none]`

Read default options from given file only.

- `--defaults-group-suffix`

Command-Line Format	`--defaults-group-suffix=string`
Type	String
Default Value	`[none]`

Also read groups with concat(group, suffix).

- `--delay=seconds`

Command-Line Format	`--delay=#`
Type	Numeric
Default Value	`5`
Minimum Value	`0`
Maximum Value	`MAX_INT`

This option sets the number of seconds to wait between executing loops. Has no effect if `--loops` is set to 0 or 1.

- `--help`

Command-Line Format	`--help`

Display help text and exit.

- `--login-path`

Command-Line Format	`--login-path=path`
Type	String
Default Value	`[none]`

Read given path from login file.

- `--loops=number`, `-l number`

Command-Line Format	`--loops=#`
Type	Numeric
Default Value	`1`
Minimum Value	`0`
Maximum Value	`MAX_INT`

This option sets the number of times to execute the select. Use `--delay` to set the time between loops.

- `--ndb-connectstring`

Command-Line Format	`--ndb-connectstring=connection-string`
Type	String
Default Value	`[none]`

Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;][host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf.

- `--ndb-mgmd-host`

Command-Line Format	`--ndb-mgmd-host=connection-string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--ndb-nodeid`

Command-Line Format	`--ndb-nodeid=#`
Type	Integer
Default Value	`[none]`

Set node ID for this node, overriding any ID set by --ndb-connectstring.

- `--ndb-optimized-node-selection`

Command-Line Format	`--ndb-optimized-node-selection`

Removed	8.0.31

Enable optimizations for selection of nodes for transactions. Enabled by default; use `--skip-ndb-optimized-node-selection` to disable.

- `--no-defaults`

Command-Line Format	`--no-defaults`

Do not read default options from any option file other than login file.

- `--print-defaults`

Command-Line Format	`--print-defaults`

Print program argument list and exit.

- `--usage`

Command-Line Format	`--usage`

Display help text and exit; same as --help.

- `--version`

Command-Line Format	`--version`

Display version information and exit.

`ndbinfo_select_all` is unable to read the following tables:

- `arbitrator_validity_detail`
- `arbitrator_validity_summary`
- `cluster_locks`
- `cluster_operations`
- `cluster_transactions`
- `disk_write_speed_aggregate_node`
- `locks_per_fragment`
- `memory_per_fragment`
- `memoryusage`
- `operations_per_fragment`
- `server_locks`
- `server_operations`
- `server_transactions`
- `table_info`

23.5.3 ndbmtd — The NDB Cluster Data Node Daemon (Multi-Threaded)

`ndbmtd` is a multithreaded version of `ndbd`, the process that is used to handle all the data in tables using the `NDBCLUSTER` storage engine. `ndbmtd` is intended for use on host computers having multiple

CPU cores. Except where otherwise noted, `ndbmtd` functions in the same way as `ndbd`; therefore, in this section, we concentrate on the ways in which `ndbmtd` differs from `ndbd`, and you should consult Section 23.5.1, “ndbd — The NDB Cluster Data Node Daemon”, for additional information about running NDB Cluster data nodes that apply to both the single-threaded and multithreaded versions of the data node process.

Command-line options and configuration parameters used with `ndbd` also apply to `ndbmtd`. For more information about these options and parameters, see Section 23.5.1, “ndbd — The NDB Cluster Data Node Daemon”, and Section 23.4.3.6, “Defining NDB Cluster Data Nodes”, respectively.

`ndbmtd` is also file system-compatible with `ndbd`. In other words, a data node running `ndbd` can be stopped, the binary replaced with `ndbmtd`, and then restarted without any loss of data. (However, when doing this, you must make sure that `MaxNoOfExecutionThreads` is set to an appropriate value before restarting the node if you wish for `ndbmtd` to run in multithreaded fashion.) Similarly, an `ndbmtd` binary can be replaced with `ndbd` simply by stopping the node and then starting `ndbd` in place of the multithreaded binary. It is not necessary when switching between the two to start the data node binary using `--initial`.

Using `ndbmtd` differs from using `ndbd` in two key respects:

1. Because `ndbmtd` runs by default in single-threaded mode (that is, it behaves like `ndbd`), you must configure it to use multiple threads. This can be done by setting an appropriate value in the `config.ini` file for the `MaxNoOfExecutionThreads` configuration parameter or the `ThreadConfig` configuration parameter. Using `MaxNoOfExecutionThreads` is simpler, but `ThreadConfig` offers more flexibility. For more information about these configuration parameters and their use, see Multi-Threading Configuration Parameters (ndbmtd).

2. Trace files are generated by critical errors in `ndbmtd` processes in a somewhat different fashion from how these are generated by `ndbd` failures. These differences are discussed in more detail in the next few paragraphs.

Like `ndbd`, `ndbmtd` generates a set of log files which are placed in the directory specified by `DataDir` in the `config.ini` configuration file. Except for trace files, these are generated in the same way and have the same names as those generated by `ndbd`.

In the event of a critical error, `ndbmtd` generates trace files describing what happened just prior to the error' occurrence. These files, which can be found in the data node's `DataDir`, are useful for analysis of problems by the NDB Cluster Development and Support teams. One trace file is generated for each `ndbmtd` thread. The names of these files have the following pattern:

```
ndb_node_id_trace.log.trace_id_tthread_id,
```

In this pattern, *`node_id`* stands for the data node's unique node ID in the cluster, *`trace_id`* is a trace sequence number, and *`thread_id`* is the thread ID. For example, in the event of the failure of an `ndbmtd` process running as an NDB Cluster data node having the node ID 3 and with `MaxNoOfExecutionThreads` equal to 4, four trace files are generated in the data node's data directory. If the is the first time this node has failed, then these files are named `ndb_3_trace.log.1_t1`, `ndb_3_trace.log.1_t2`, `ndb_3_trace.log.1_t3`, and `ndb_3_trace.log.1_t4`. Internally, these trace files follow the same format as `ndbd` trace files.

The `ndbd` exit codes and messages that are generated when a data node process shuts down prematurely are also used by `ndbmtd`. See Data Node Error Messages, for a listing of these.

Note

It is possible to use `ndbd` and `ndbmtd` concurrently on different data nodes in the same NDB Cluster. However, such configurations have not been tested extensively; thus, we cannot recommend doing so in a production setting at this time.

23.5.4 ndb_mgmd — The NDB Cluster Management Server Daemon

The management server is the process that reads the cluster configuration file and distributes this information to all nodes in the cluster that request it. It also maintains a log of cluster activities. Management clients can connect to the management server and check the cluster's status.

All options that can be used with `ndb_mgmd` are shown in the following table. Additional descriptions follow the table.

Table 23.26 Command-line options used with the program ndb_mgmd

Format	Description	Added, Deprecated, or Removed
`--bind-address=host`	Local bind address	(Supported in all NDB releases based on MySQL 8.0)
`--character-sets-dir=path`	Directory containing character sets	REMOVED: 8.0.31
`--cluster-config-suffix=name`	Override defaults group suffix when reading cluster_config sections in my.cnf file; used in testing	ADDED: NDB 8.0.24
`--config-cache[=TRUE|FALSE]`	Enable management server configuration cache; true by default	(Supported in all NDB releases based on MySQL 8.0)
`--config-file=file`, `-f file`	Specify cluster configuration file; also specify --reload or --initial to override configuration cache if present	(Supported in all NDB releases based on MySQL 8.0)
`--configdir=directory`, `--config-dir=directory`	Specify cluster management server configuration cache directory	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retries=#`	Number of times to retry connection before giving up	REMOVED: 8.0.31
`--connect-retry-delay=#`	Number of seconds to wait between attempts to contact management server	REMOVED: 8.0.31
`--connect-string=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--core-file`	Write core file on error; used in debugging	REMOVED: 8.0.31
`--daemon`, `-d`	Run ndb_mgmd in daemon mode (default)	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-extra-file=path`	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-file=path`	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-group-suffix=string`	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--initial`	Causes management server to reload configuration data from	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
	configuration file, bypassing configuration cache	
`--install[=name]`	Used to install management server process as Windows service; does not apply on other platforms	(Supported in all NDB releases based on MySQL 8.0)
`--interactive`	Run ndb_mgmd in interactive mode (not officially supported in production; for testing purposes only)	(Supported in all NDB releases based on MySQL 8.0)
`--log-name=name`	Name to use when writing cluster log messages applying to this node	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)
`--mycnf`	Read cluster configuration data from my.cnf file	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-connectstring=connection_string`, `-c connection_string`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-mgmd-host=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable	REMOVED: 8.0.31
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--no-nodeid-checks`	Do not perform any node ID checks	(Supported in all NDB releases based on MySQL 8.0)
`--nodaemon`	Do not run ndb_mgmd as a daemon	(Supported in all NDB releases based on MySQL 8.0)
`--nowait-nodes=list`	Do not wait for management nodes specified when starting this management server; requires --ndb-nodeid option	(Supported in all NDB releases based on MySQL 8.0)
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
`--print-full-config`, `-P`	Print full configuration and exit	(Supported in all NDB releases based on MySQL 8.0)
`--reload`	Causes management server to compare configuration file with configuration cache	(Supported in all NDB releases based on MySQL 8.0)
`--remove[=name]`	Used to remove management server process that was previously installed as Windows service, optionally specifying name of service to be removed; does not apply on other platforms	(Supported in all NDB releases based on MySQL 8.0)
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--skip-config-file`	Do not use configuration file	(Supported in all NDB releases based on MySQL 8.0)
`--verbose`, `-v`	Write additional information to log	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)

- `--bind-address=host`

Command-Line Format	`--bind-address=host`
Type	String
Default Value	`[none]`

Causes the management server to bind to a specific network interface (host name or IP address). This option has no default value.

- `--character-sets-dir`

Command-Line Format	`--character-sets-dir=path`
Removed	8.0.31

Directory containing character sets.

- `cluster-config-suffix`

Command-Line Format	`--cluster-config-suffix=name`
Introduced	8.0.24-ndb-8.0.24
Type	String
Default Value	`[none]`

Override defaults group suffix when reading cluster configuration sections in `my.cnf`; used in testing.

- `--config-cache`

Command-Line Format	`--config-cache[=TRUE|FALSE]`
Type	Boolean
Default Value	`TRUE`

This option, whose default value is `1` (or `TRUE`, or `ON`), can be used to disable the management server's configuration cache, so that it reads its configuration from `config.ini` every time it starts (see Section 23.4.3, "NDB Cluster Configuration Files"). You can do this by starting the `ndb_mgmd` process with any one of the following options:

- `--config-cache=0`
- `--config-cache=FALSE`
- `--config-cache=OFF`
- `--skip-config-cache`

Using one of the options just listed is effective only if the management server has no stored configuration at the time it is started. If the management server finds any configuration cache files, then the `--config-cache` option or the `--skip-config-cache` option is ignored. Therefore, to disable configuration caching, the option should be used the *first* time that the management server is started. Otherwise—that is, if you wish to disable configuration caching for a management server that has *already* created a configuration cache—you must stop the management server, delete any existing configuration cache files manually, then restart the management server with `--skip-config-cache` (or with `--config-cache` set equal to 0, `OFF`, or `FALSE`).

Configuration cache files are normally created in a directory named `mysql-cluster` under the installation directory (unless this location has been overridden using the `--configdir` option). Each time the management server updates its configuration data, it writes a new cache file. The files are named sequentially in order of creation using the following format:

```
ndb_node-id_config.bin.seq-number
```

`node-id` is the management server's node ID; *`seq-number`* is a sequence number, beginning with 1. For example, if the management server's node ID is 5, then the first three configuration cache files would, when they are created, be named `ndb_5_config.bin.1`, `ndb_5_config.bin.2`, and `ndb_5_config.bin.3`.

If your intent is to purge or reload the configuration cache without actually disabling caching, you should start `ndb_mgmd` with one of the options `--reload` or `--initial` instead of `--skip-config-cache`.

To re-enable the configuration cache, simply restart the management server, but without the `--config-cache` or `--skip-config-cache` option that was used previously to disable the configuration cache.

`ndb_mgmd` does not check for the configuration directory (`--configdir`) or attempts to create one when `--skip-config-cache` is used. (Bug #13428853)

- `--config-file=filename`, `-f filename`

Command-Line Format	`--config-file=file`
Disabled by	`skip-config-file`
Type	File name

Default Value	[none]

Instructs the management server as to which file it should use for its configuration file. By default, the management server looks for a file named config.ini in the same directory as the ndb_mgmd executable; otherwise the file name and location must be specified explicitly.

This option has no default value, and is ignored unless the management server is forced to read the configuration file, either because ndb_mgmd was started with the --reload or --initial option, or because the management server could not find any configuration cache. Beginning with NDB 8.0.26, ndb_mgmd refuses to start if --config-file is specified without either of --initial or --reload.

The --config-file option is also read if ndb_mgmd was started with --config-cache=OFF. See Section 23.4.3, “NDB Cluster Configuration Files”, for more information.

- --configdir=*dir_name*

Command-Line Format	--configdir=directory --config-dir=directory
Type	File name
Default Value	$INSTALLDIR/mysql-cluster

Specifies the cluster management server's configuration cache directory. --config-dir is an alias for this option.

In NDB 8.0.27 and later, this must be an absolute path. Otherwise, the management server refuses to start.

- --connect-retries

Command-Line Format	--connect-retries=#
Removed	8.0.31
Type	Integer
Default Value	12
Minimum Value	0
Maximum Value	12

Number of times to retry connection before giving up.

- --connect-retry-delay

Command-Line Format	--connect-retry-delay=#
Removed	8.0.31
Type	Integer
Default Value	5
Minimum Value	0
Maximum Value	5

Number of seconds to wait between attempts to contact management server.

- --connect-string

Command-Line Format	--connect-string=connection_string

Type	String
Default Value	[none]

Same as --ndb-connectstring.

- --core-file

Command-Line Format	--core-file
Removed	8.0.31

Write core file on error; used in debugging.

- --daemon, -d

Command-Line Format	--daemon

Instructs ndb_mgmd to start as a daemon process. This is the default behavior.

This option has no effect when running ndb_mgmd on Windows platforms.

- --defaults-extra-file

Command-Line Format	--defaults-extra-file=path
Type	String
Default Value	[none]

Read given file after global files are read.

- --defaults-file

Command-Line Format	--defaults-file=path
Type	String
Default Value	[none]

Read default options from given file only.

- --defaults-group-suffix

Command-Line Format	--defaults-group-suffix=string
Type	String
Default Value	[none]

Also read groups with concat(group, suffix).

- --help

Command-Line Format	--help

Display help text and exit.

- --initial

Command-Line Format	--initial

Configuration data is cached internally, rather than being read from the cluster global configuration file each time the management server is started (see Section 23.4.3, “NDB Cluster Configuration

Files"). Using the `--initial` option overrides this behavior, by forcing the management server to delete any existing cache files, and then to re-read the configuration data from the cluster configuration file and to build a new cache.

This differs in two ways from the `--reload` option. First, `--reload` forces the server to check the configuration file against the cache and reload its data only if the contents of the file are different from the cache. Second, `--reload` does not delete any existing cache files.

If `ndb_mgmd` is invoked with `--initial` but cannot find a global configuration file, the management server cannot start.

When a management server starts, it checks for another management server in the same NDB Cluster and tries to use the other management server's configuration data. This behavior has implications when performing a rolling restart of an NDB Cluster with multiple management nodes. See Section 23.6.5, "Performing a Rolling Restart of an NDB Cluster", for more information.

When used together with the `--config-file` option, the cache is cleared only if the configuration file is actually found.

- `--install[=name]`

Command-Line Format	`--install[=name]`
Platform Specific	Windows
Type	String
Default Value	`ndb_mgmd`

Causes `ndb_mgmd` to be installed as a Windows service. Optionally, you can specify a name for the service; if not set, the service name defaults to `ndb_mgmd`. Although it is preferable to specify other `ndb_mgmd` program options in a `my.ini` or `my.cnf` configuration file, it is possible to use them together with `--install`. However, in such cases, the `--install` option must be specified first, before any other options are given, for the Windows service installation to succeed.

It is generally not advisable to use this option together with the `--initial` option, since this causes the configuration cache to be wiped and rebuilt every time the service is stopped and started. Care should also be taken if you intend to use any other `ndb_mgmd` options that affect the starting of the management server, and you should make absolutely certain you fully understand and allow for any possible consequences of doing so.

The `--install` option has no effect on non-Windows platforms.

- `--interactive`

Command-Line Format	`--interactive`

Starts `ndb_mgmd` in interactive mode; that is, an `ndb_mgm` client session is started as soon as the management server is running. This option does not start any other NDB Cluster nodes.

- `--log-name=name`

Command-Line Format	`--log-name=name`
Type	String
Default Value	`MgmtSrvr`

Provides a name to be used for this node in the cluster log.

- `--login-path`

Command-Line Format	`--login-path=path`

Type	String
Default Value	[none]

Read given path from login file.

- --mycnf

Command-Line Format	--mycnf

Read configuration data from the my.cnf file.

- --ndb-connectstring

Command-Line Format	--ndb-connectstring=connection_string
Type	String
Default Value	[none]

Set connection string. Syntax: [nodeid=*id*;][host=]*hostname*[:*port*]. Overrides entries in NDB_CONNECTSTRING and my.cnf. Ignored if --config-file is specified; beginning with NDB 8.0.27, a warning is issued when both options are used.

- --ndb-mgmd-host

Command-Line Format	--ndb-mgmd-host=connection_string
Type	String
Default Value	[none]

Same as --ndb-connectstring.

- --ndb-nodeid

Command-Line Format	--ndb-nodeid=#
Type	Integer
Default Value	[none]

Set node ID for this node, overriding any ID set by --ndb-connectstring.

- --ndb-optimized-node-selection

Command-Line Format	--ndb-optimized-node-selection
Removed	8.0.31

Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable.

- --no-nodeid-checks

Command-Line Format	--no-nodeid-checks

Do not perform any checks of node IDs.

- `--nodaemon`

Command-Line Format	`--nodaemon`

Instructs `ndb_mgmd` not to start as a daemon process.

The default behavior for `ndb_mgmd` on Windows is to run in the foreground, making this option unnecessary on Windows platforms.

- `--no-defaults`

Command-Line Format	`--no-defaults`

Do not read default options from any option file other than login file.

- `--nowait-nodes`

Command-Line Format	`--nowait-nodes=list`
Type	Numeric
Default Value	`[none]`
Minimum Value	`1`
Maximum Value	`255`

When starting an NDB Cluster is configured with two management nodes, each management server normally checks to see whether the other `ndb_mgmd` is also operational and whether the other management server's configuration is identical to its own. However, it is sometimes desirable to start the cluster with only one management node (and perhaps to allow the other `ndb_mgmd` to be started later). This option causes the management node to bypass any checks for any other management nodes whose node IDs are passed to this option, permitting the cluster to start as though configured to use only the management node that was started.

For purposes of illustration, consider the following portion of a `config.ini` file (where we have omitted most of the configuration parameters that are not relevant to this example):

```
[ndbd]
NodeId = 1
HostName = 198.51.100.101

[ndbd]
NodeId = 2
HostName = 198.51.100.102

[ndbd]
NodeId = 3
HostName = 198.51.100.103

[ndbd]
NodeId = 4
HostName = 198.51.100.104

[ndb_mgmd]
NodeId = 10
HostName = 198.51.100.150

[ndb_mgmd]
NodeId = 11
HostName = 198.51.100.151

[api]
NodeId = 20
HostName = 198.51.100.200

[api]
```

```
NodeId = 21
HostName = 198.51.100.201
```

Assume that you wish to start this cluster using only the management server having node ID `10` and running on the host having the IP address 198.51.100.150. (Suppose, for example, that the host computer on which you intend to the other management server is temporarily unavailable due to a hardware failure, and you are waiting for it to be repaired.) To start the cluster in this way, use a command line on the machine at 198.51.100.150 to enter the following command:

```
$> ndb_mgmd --ndb-nodeid=10 --nowait-nodes=11
```

As shown in the preceding example, when using `--nowait-nodes`, you must also use the `--ndb-nodeid` option to specify the node ID of this `ndb_mgmd` process.

You can then start each of the cluster's data nodes in the usual way. If you wish to start and use the second management server in addition to the first management server at a later time without restarting the data nodes, you must start each data node with a connection string that references both management servers, like this:

```
$> ndbd -c 198.51.100.150,198.51.100.151
```

The same is true with regard to the connection string used with any `mysqld` processes that you wish to start as NDB Cluster SQL nodes connected to this cluster. See Section 23.4.3.3, “NDB Cluster Connection Strings”, for more information.

When used with `ndb_mgmd`, this option affects the behavior of the management node with regard to other management nodes only. Do not confuse it with the `--nowait-nodes` option used with `ndbd` or `ndbmtd` to permit a cluster to start with fewer than its full complement of data nodes; when used with data nodes, this option affects their behavior only with regard to other data nodes.

Multiple management node IDs may be passed to this option as a comma-separated list. Each node ID must be no less than 1 and no greater than 255. In practice, it is quite rare to use more than two management servers for the same NDB Cluster (or to have any need for doing so); in most cases you need to pass to this option only the single node ID for the one management server that you do not wish to use when starting the cluster.

Note

When you later start the “missing” management server, its configuration must match that of the management server that is already in use by the cluster. Otherwise, it fails the configuration check performed by the existing management server, and does not start.

- `--print-defaults`

Command-Line Format	`--print-defaults`

Print program argument list and exit.

- `--print-full-config`, `-P`

Command-Line Format	`--print-full-config`

Shows extended information regarding the configuration of the cluster. With this option on the command line the `ndb_mgmd` process prints information about the cluster setup including an extensive list of the cluster configuration sections as well as parameters and their values. Normally used together with the `--config-file` (`-f`) option.

- `--reload`

Command-Line Format	`--reload`

NDB Cluster configuration data is stored internally rather than being read from the cluster global configuration file each time the management server is started (see Section 23.4.3, “NDB Cluster Configuration Files”). Using this option forces the management server to check its internal data store against the cluster configuration file and to reload the configuration if it finds that the configuration file does not match the cache. Existing configuration cache files are preserved, but not used.

This differs in two ways from the `--initial` option. First, `--initial` causes all cache files to be deleted. Second, `--initial` forces the management server to re-read the global configuration file and construct a new cache.

If the management server cannot find a global configuration file, then the `--reload` option is ignored.

When `--reload` is used, the management server must be able to communicate with data nodes and any other management servers in the cluster before it attempts to read the global configuration file; otherwise, the management server fails to start. This can happen due to changes in the networking environment, such as new IP addresses for nodes or an altered firewall configuration. In such cases, you must use `--initial` instead to force the existing cached configuration to be discarded and reloaded from the file. See Section 23.6.5, “Performing a Rolling Restart of an NDB Cluster”, for additional information.

- `--remove[=name]`

Command-Line Format	`--remove[=name]`
Platform Specific	Windows
Type	String
Default Value	`ndb_mgmd`

Remove a management server process that has been installed as a Windows service, optionally specifying the name of the service to be removed. Applies only to Windows platforms.

- `--skip-config-file`

Command-Line Format	`--skip-config-file`

Do not read cluster configuration file; ignore `--initial` and `--reload` options if specified.

- `--usage`

Command-Line Format	`--usage`

Display help text and exit; same as --help.

- `--verbose`, `-v`

Command-Line Format	`--verbose`

Remove a management server process that has been installed as a Windows service, optionally specifying the name of the service to be removed. Applies only to Windows platforms.

- `--version`

Command-Line Format	`--version`

Display version information and exit.

It is not strictly necessary to specify a connection string when starting the management server. However, if you are using more than one management server, a connection string should be provided and each node in the cluster should specify its node ID explicitly.

See Section 23.4.3.3, “NDB Cluster Connection Strings”, for information about using connection strings. Section 23.5.4, “ndb_mgmd — The NDB Cluster Management Server Daemon”, describes other options for `ndb_mgmd`.

The following files are created or used by `ndb_mgmd` in its starting directory, and are placed in the `DataDir` as specified in the `config.ini` configuration file. In the list that follows, *`node_id`* is the unique node identifier.

- `config.ini` is the configuration file for the cluster as a whole. This file is created by the user and read by the management server. Section 23.4, “Configuration of NDB Cluster”, discusses how to set up this file.
- `ndb_`*`node_id`*`_cluster.log` is the cluster events log file. Examples of such events include checkpoint startup and completion, node startup events, node failures, and levels of memory usage. A complete listing of cluster events with descriptions may be found in Section 23.6, “Management of NDB Cluster”.

  By default, when the size of the cluster log reaches one million bytes, the file is renamed to `ndb_`*`node_id`*`_cluster.log.`*`seq_id`*, where *`seq_id`* is the sequence number of the cluster log file. (For example: If files with the sequence numbers 1, 2, and 3 already exist, the next log file is named using the number `4`.) You can change the size and number of files, and other characteristics of the cluster log, using the `LogDestination` configuration parameter.
- `ndb_`*`node_id`*`_out.log` is the file used for `stdout` and `stderr` when running the management server as a daemon.
- `ndb_`*`node_id`*`.pid` is the process ID file used when running the management server as a daemon.

23.5.5 ndb_mgm — The NDB Cluster Management Client

The `ndb_mgm` management client process is actually not needed to run the cluster. Its value lies in providing a set of commands for checking the cluster's status, starting backups, and performing other administrative functions. The management client accesses the management server using a C API. Advanced users can also employ this API for programming dedicated management processes to perform tasks similar to those performed by `ndb_mgm`.

To start the management client, it is necessary to supply the host name and port number of the management server:

```
$> ndb_mgm [host_name [port_num]]
```

For example:

```
$> ndb_mgm ndb_mgmd.mysql.com 1186
```

The default host name and port number are `localhost` and 1186, respectively.

All options that can be used with `ndb_mgm` are shown in the following table. Additional descriptions follow the table.

Table 23.27 Command-line options used with the program ndb_mgm

Format	Description	Added, Deprecated, or Removed
`--backup-password-from-stdin`	Get decryption password in a secure fashion from STDIN; use together with --execute and ndb_mgm START BACKUP command	ADDED: NDB 8.0.24

Format	Description	Added, Deprecated, or Removed
`--character-sets-dir=path`	Directory containing character sets	REMOVED: 8.0.31
`--connect-retries=#`	Set number of times to retry connection before giving up; 0 means 1 attempt only (and no retries)	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retry-delay=#`	Number of seconds to wait between attempts to contact management server	(Supported in all NDB releases based on MySQL 8.0)
`--connect-string=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--core-file`	Write core file on error; used in debugging	REMOVED: 8.0.31
`--defaults-extra-file=path`	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-file=path`	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-group-suffix=string`	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)
`--encrypt-backup`	Cause START BACKUP to encrypt whenever making a backup, prompting for password if not supplied by user	ADDED: NDB 8.0.24
`--execute=command`, `-e command`	Execute command and exit	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-connectstring=connection_string`, `-c connection_string`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-mgmd-host=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default;	REMOVED: 8.0.31

Format	Description	Added, Deprecated, or Removed
	use --skip-ndb-optimized-node-selection to disable	
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)
`--try-reconnect=#`, `-t #`	Set number of times to retry connection before giving up; synonym for --connect-retries	(Supported in all NDB releases based on MySQL 8.0)
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)

- `--backup-password-from-stdin[=TRUE|FALSE]`

Command-Line Format	`--backup-password-from-stdin`
Introduced	8.0.24-ndb-8.0.24

This option enables input of the backup password from the system shell (`stdin`) when using `--execute "START BACKUP"` or similar to create a backup. Use of this option requires use of `--execute` as well.

- `--character-sets-dir`

Command-Line Format	`--character-sets-dir=path`
Removed	8.0.31

Directory containing character sets.

- `--connect-retries=#`

Command-Line Format	`--connect-retries=#`
Type	Numeric
Default Value	`3`
Minimum Value	`0`
Maximum Value	`4294967295`

This option specifies the number of times following the first attempt to retry a connection before giving up (the client always tries the connection at least once). The length of time to wait per attempt is set using `--connect-retry-delay`.

This option is synonymous with the `--try-reconnect` option, which is now deprecated.

- `--connect-retry-delay`

Command-Line Format	`--connect-retry-delay=#`
Type	Integer
Default Value	`5`

Minimum Value	0
Maximum Value	5

Number of seconds to wait between attempts to contact management server.

- `--connect-string`

Command-Line Format	`--connect-string=connection_string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--core-file`

Command-Line Format	`--core-file`
Removed	8.0.31

Write core file on error; used in debugging.

- `--defaults-extra-file`

Command-Line Format	`--defaults-extra-file=path`
Type	String
Default Value	`[none]`

Read given file after global files are read.

- `--defaults-file`

Command-Line Format	`--defaults-file=path`
Type	String
Default Value	`[none]`

Read default options from given file only.

- `--defaults-group-suffix`

Command-Line Format	`--defaults-group-suffix=string`
Type	String
Default Value	`[none]`

Also read groups with concat(group, suffix).

- `--encrypt-backup`

Command-Line Format	`--encrypt-backup`
Introduced	8.0.24-ndb-8.0.24

When used, this option causes all backups to be encrypted. To make this happen whenever `ndb_mgm` is run, put the option in the `[ndb_mgm]` section of the `my.cnf` file.

- `--execute=command`, `-e command`

Command-Line Format	`--execute=command`

This option can be used to send a command to the NDB Cluster management client from the system shell. For example, either of the following is equivalent to executing `SHOW` in the management client:

```
$> ndb_mgm -e "SHOW"

$> ndb_mgm --execute="SHOW"
```

This is analogous to how the `--execute` or `-e` option works with the `mysql` command-line client. See Section 4.2.2.1, “Using Options on the Command Line”.

Note

If the management client command to be passed using this option contains any space characters, then the command *must* be enclosed in quotation marks. Either single or double quotation marks may be used. If the management client command contains no space characters, the quotation marks are optional.

- `--help`

Command-Line Format	`--help`

Display help text and exit.

- `--login-path`

Command-Line Format	`--login-path=path`
Type	String
Default Value	`[none]`

Read given path from login file.

- `--ndb-connectstring`

Command-Line Format	`--ndb-connectstring=connection_string`
Type	String
Default Value	`[none]`

Set connect string for connecting to `ndb_mgmd`. Syntax: [`nodeid=`*`id`*`;`][`host=`]*`hostname`*[`:`*`port`*]. Overrides entries in `NDB_CONNECTSTRING` and `my.cnf`.

- `--ndb-nodeid`

Command-Line Format	`--ndb-nodeid=#`
Type	Integer
Default Value	`[none]`

Set node ID for this node, overriding any ID set by `--ndb-connectstring`.

- `--ndb-mgmd-host`

Command-Line Format	`--ndb-mgmd-host=connection_string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--ndb-optimized-node-selection`

Command-Line Format	`--ndb-optimized-node-selection`
Removed	8.0.31

Enable optimizations for selection of nodes for transactions. Enabled by default; use `--skip-ndb-optimized-node-selection` to disable.

- `--no-defaults`

Command-Line Format	`--no-defaults`

Do not read default options from any option file other than login file.

- `--print-defaults`

Command-Line Format	`--print-defaults`

Print program argument list and exit.

- `--try-reconnect=`*`number`*

Command-Line Format	`--try-reconnect=#`
Deprecated	Yes
Type	Numeric
Type	Integer
Default Value	`12`
Default Value	`3`
Minimum Value	`0`
Maximum Value	`4294967295`

If the connection to the management server is broken, the node tries to reconnect to it every 5 seconds until it succeeds. By using this option, it is possible to limit the number of attempts to *`number`* before giving up and reporting an error instead.

This option is deprecated and subject to removal in a future release. Use `--connect-retries`, instead.

- `--usage`

Command-Line Format	`--usage`

Display help text and exit; same as `--help`.

- `--version`

Command-Line Format	`--version`

Display version information and exit.

Additional information about using `ndb_mgm` can be found in Section 23.6.1, “Commands in the NDB Cluster Management Client”.

23.5.6 ndb_blob_tool — Check and Repair BLOB and TEXT columns of NDB Cluster Tables

This tool can be used to check for and remove orphaned BLOB column parts from NDB tables, as well as to generate a file listing any orphaned parts. It is sometimes useful in diagnosing and repairing corrupted or damaged NDB tables containing BLOB or TEXT columns.

The basic syntax for ndb_blob_tool is shown here:

```
ndb_blob_tool [options] table [column, ...]
```

Unless you use the --help option, you must specify an action to be performed by including one or more of the options --check-orphans, --delete-orphans, or --dump-file. These options cause ndb_blob_tool to check for orphaned BLOB parts, remove any orphaned BLOB parts, and generate a dump file listing orphaned BLOB parts, respectively, and are described in more detail later in this section.

You must also specify the name of a table when invoking ndb_blob_tool. In addition, you can optionally follow the table name with the (comma-separated) names of one or more BLOB or TEXT columns from that table. If no columns are listed, the tool works on all of the table's BLOB and TEXT columns. If you need to specify a database, use the --database (-d) option.

The --verbose option provides additional information in the output about the tool's progress.

All options that can be used with ndb_mgmd are shown in the following table. Additional descriptions follow the table.

Table 23.28 Command-line options used with the program ndb_blob_tool

Format	Description	Added, Deprecated, or Removed
--add-missing	Write dummy blob parts to take place of those which are missing	ADDED: NDB 8.0.20
--character-sets-dir=path	Directory containing character sets	REMOVED: 8.0.31
--check-missing	Check for blobs having inline parts but missing one or more parts from parts table	ADDED: NDB 8.0.20
--check-orphans	Check for blob parts having no corresponding inline parts	(Supported in all NDB releases based on MySQL 8.0)
--connect-retries=#	Number of times to retry connection before giving up	(Supported in all NDB releases based on MySQL 8.0)
--connect-retry-delay=#	Number of seconds to wait between attempts to contact management server	(Supported in all NDB releases based on MySQL 8.0)
--connect-string=connection_string, -c connection_string	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
--core-file	Write core file on error; used in debugging	REMOVED: 8.0.31
--database=name, -d name	Database to find the table in	(Supported in all NDB releases based on MySQL 8.0)
--defaults-extra-file=path	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
--defaults-file=path	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
--defaults-group-suffix=string	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
`--delete-orphans`	Delete blob parts having no corresponding inline parts	(Supported in all NDB releases based on MySQL 8.0)
`--dump-file=file`	Write orphan keys to specified file	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-connectstring=connection_string`, `-c connection_string`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-mgmd-host=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable	REMOVED: 8.0.31
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--verbose`, `-v`	Verbose output	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)

- `--add-missing`

Command-Line Format	`--add-missing`
Introduced	8.0.20-ndb-8.0.20

For each inline part in NDB Cluster tables which has no corresponding BLOB part, write a dummy BLOB part of the required length, consisting of spaces.

- `--character-sets-dir`

Command-Line Format	`--character-sets-dir=path`
Removed	8.0.31

Directory containing character sets.

- `--check-missing`

Command-Line Format	`--check-missing`
Introduced	8.0.20-ndb-8.0.20

Check for inline parts in NDB Cluster tables which have no corresponding BLOB parts.

- `--check-orphans`

Command-Line Format	`--check-orphans`

Check for BLOB parts in NDB Cluster tables which have no corresponding inline parts.

- `--connect-retries`

Command-Line Format	`--connect-retries=#`
Type	Integer
Default Value	`12`
Minimum Value	`0`
Maximum Value	`12`

Number of times to retry connection before giving up.

- `--connect-retry-delay`

Command-Line Format	`--connect-retry-delay=#`
Type	Integer
Default Value	`5`
Minimum Value	`0`
Maximum Value	`5`

Number of seconds to wait between attempts to contact management server.

- `--connect-string`

Command-Line Format	`--connect-string=connection_string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--core-file`

Command-Line Format	`--core-file`
Removed	8.0.31

Write core file on error; used in debugging.

- `--database=db_name`, `-d`

Command-Line Format	`--database=name`
Type	String
Default Value	`[none]`

Specify the database to find the table in.

- `--defaults-extra-file`

Command-Line Format	`--defaults-extra-file=path`
Type	String
Default Value	`[none]`

Read given file after global files are read.

- `--defaults-file`

Command-Line Format	`--defaults-file=path`
Type	String
Default Value	`[none]`

Read default options from given file only.

- `--defaults-group-suffix`

Command-Line Format	`--defaults-group-suffix=string`
Type	String
Default Value	`[none]`

Also read groups with concat(group, suffix).

- `--delete-orphans`

Command-Line Format	`--delete-orphans`

Remove BLOB parts from NDB Cluster tables which have no corresponding inline parts.

- `--dump-file=file`

Command-Line Format	`--dump-file=file`
Type	File name
Default Value	`[none]`

Writes a list of orphaned BLOB column parts to *`file`*. The information written to the file includes the table key and BLOB part number for each orphaned BLOB part.

- `--help`

Command-Line Format	`--help`

Display help text and exit.

- `--login-path`

Command-Line Format	`--login-path=path`

Type	String
Default Value	[none]

Read given path from login file.

- --ndb-connectstring

Command-Line Format	--ndb-connectstring=connection_string
Type	String
Default Value	[none]

Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;][host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf.

- --ndb-mgmd-host

Command-Line Format	--ndb-mgmd-host=connection_string
Type	String
Default Value	[none]

Same as --ndb-connectstring.

- --ndb-nodeid

Command-Line Format	--ndb-nodeid=#
Type	Integer
Default Value	[none]

Set node ID for this node, overriding any ID set by --ndb-connectstring.

- --ndb-optimized-node-selection

Command-Line Format	--ndb-optimized-node-selection
Removed	8.0.31

Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable.

- --no-defaults

Command-Line Format	--no-defaults

Do not read default options from any option file other than login file.

- --print-defaults

Command-Line Format	--print-defaults

Print program argument list and exit.

- --usage

Command-Line Format	--usage

Display help text and exit; same as --help.

- `--verbose`

Command-Line Format	`--verbose`

Provide extra information in the tool's output regarding its progress.

- `--version`

Command-Line Format	`--version`

Display version information and exit.

Example

First we create an `NDB` table in the `test` database, using the `CREATE TABLE` statement shown here:

```
USE test;

CREATE TABLE btest (
    c0 BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    c1 TEXT,
    c2 BLOB
)   ENGINE=NDB;
```

Then we insert a few rows into this table, using a series of statements similar to this one:

```
INSERT INTO btest VALUES (NULL, 'x', REPEAT('x', 1000));
```

When run with `--check-orphans` against this table, `ndb_blob_tool` generates the following output:

```
$> ndb_blob_tool --check-orphans --verbose -d test btest
connected
processing 2 blobs
processing blob #0 c1 NDB$BLOB_19_1
NDB$BLOB_19_1: nextResult: res=1
total parts: 0
orphan parts: 0
processing blob #1 c2 NDB$BLOB_19_2
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=0
NDB$BLOB_19_2: nextResult: res=1
total parts: 10
orphan parts: 0
disconnected

NDBT_ProgramExit: 0 - OK
```

The tool reports that there are no `NDB` BLOB column parts associated with column `c1`, even though `c1` is a `TEXT` column. This is due to the fact that, in an `NDB` table, only the first 256 bytes of a `BLOB` or `TEXT` column value are stored inline, and only the excess, if any, is stored separately; thus, if there are no values using more than 256 bytes in a given column of one of these types, no `BLOB` column parts are created by `NDB` for this column. See Section 11.7, “Data Type Storage Requirements”, for more information.

23.5.7 ndb_config — Extract NDB Cluster Configuration Information

This tool extracts current configuration information for data nodes, SQL nodes, and API nodes from one of a number of sources: an NDB Cluster management node, or its `config.ini` or `my.cnf` file. By

default, the management node is the source for the configuration data; to override the default, execute ndb_config with the `--config-file` or `--mycnf` option. It is also possible to use a data node as the source by specifying its node ID with `--config_from_node=node_id`.

`ndb_config` can also provide an offline dump of all configuration parameters which can be used, along with their default, maximum, and minimum values and other information. The dump can be produced in either text or XML format; for more information, see the discussion of the `--configinfo` and `--xml` options later in this section).

You can filter the results by section (`DB`, `SYSTEM`, or `CONNECTIONS`) using one of the options `--nodes`, `--system`, or `--connections`.

All options that can be used with `ndb_config` are shown in the following table. Additional descriptions follow the table.

Table 23.29 Command-line options used with the program ndb_config

Format	Description	Added, Deprecated, or Removed
`--character-sets-dir=path`	Directory containing character sets	REMOVED: 8.0.31
`--cluster-config-suffix=name`	Override defaults group suffix when reading cluster_config sections in my.cnf file; used in testing	ADDED: NDB 8.0.24
`--config-binary-file=path/to/file`	Read this binary configuration file	ADDED: NDB 8.0.32
`--config-file=file_name`	Set the path to config.ini file	(Supported in all NDB releases based on MySQL 8.0)
`--config-from-node=#`	Obtain configuration data from the node having this ID (must be a data node)	(Supported in all NDB releases based on MySQL 8.0)
`--configinfo`	Dumps information about all NDB configuration parameters in text format with default, maximum, and minimum values. Use with --xml to obtain XML output	(Supported in all NDB releases based on MySQL 8.0)
`--connections`	Print information only about connections specified in [tcp], [tcp default], [sci], [sci default], [shm], or [shm default] sections of cluster configuration file. Cannot be used with --system or --nodes	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retries=#`	Number of times to retry connection before giving up	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retry-delay=#`	Number of seconds to wait between attempts to contact management server	(Supported in all NDB releases based on MySQL 8.0)
`--connect-string=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--core-file`	Write core file on error; used in debugging	REMOVED: 8.0.31

Format	Description	Added, Deprecated, or Removed
`--defaults-extra-file=path`	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-file=path`	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-group-suffix=string`	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)
`--diff-default`	Print only configuration parameters that have non-default values	(Supported in all NDB releases based on MySQL 8.0)
`--fields=string`, `-f`	Field separator	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--host=name`	Specify host	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)
`--mycnf`	Read configuration data from my.cnf file	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-connectstring=connection_string`, `-c connection_string`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-mgmd-host=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	REMOVED: 8.0.31
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable	REMOVED: 8.0.31
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--nodeid=#`	Get configuration of node with this ID	(Supported in all NDB releases based on MySQL 8.0)
`--nodes`	Print node information ([ndbd] or [ndbd default] section of cluster configuration file) only. Cannot be used with --system or --connections	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
`--query=string`, `-q string`	One or more query options (attributes)	(Supported in all NDB releases based on MySQL 8.0)
`--query-all`, `-a`	Dumps all parameters and values to a single comma-delimited string	(Supported in all NDB releases based on MySQL 8.0)
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)
`--rows=string`, `-r string`	Row separator	(Supported in all NDB releases based on MySQL 8.0)
`--system`	Print SYSTEM section information only (see ndb_config --configinfo output). Cannot be used with --nodes or --connections	(Supported in all NDB releases based on MySQL 8.0)
`--type=name`	Specify node type	(Supported in all NDB releases based on MySQL 8.0)
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)
`--configinfo --xml`	Use --xml with --configinfo to obtain a dump of all NDB configuration parameters in XML format with default, maximum, and minimum values	(Supported in all NDB releases based on MySQL 8.0)

- `cluster-config-suffix`

Command-Line Format	`--cluster-config-suffix=name`
Introduced	8.0.24-ndb-8.0.24
Type	String
Default Value	`[none]`

Override defaults group suffix when reading cluster configuration sections in `my.cnf`; used in testing.

- `--configinfo`

  The `--configinfo` option causes `ndb_config` to dump a list of each NDB Cluster configuration parameter supported by the NDB Cluster distribution of which `ndb_config` is a part, including the following information:

  - A brief description of each parameter's purpose, effects, and usage
  - The section of the `config.ini` file where the parameter may be used
  - The parameter's data type or unit of measurement
  - Where applicable, the parameter's default, minimum, and maximum values
  - NDB Cluster release version and build information

  By default, this output is in text format. Part of this output is shown here:

```
$> ndb_config --configinfo

****** SYSTEM ******

Name (String)
Name of system (NDB Cluster)
MANDATORY

PrimaryMGMNode (Non-negative Integer)
Node id of Primary ndb_mgmd(MGM) node
Default: 0 (Min: 0, Max: 4294967039)

ConfigGenerationNumber (Non-negative Integer)
Configuration generation number
Default: 0 (Min: 0, Max: 4294967039)

****** DB ******

MaxNoOfSubscriptions (Non-negative Integer)
Max no of subscriptions (default 0 == MaxNoOfTables)
Default: 0 (Min: 0, Max: 4294967039)

MaxNoOfSubscribers (Non-negative Integer)
Max no of subscribers (default 0 == 2 * MaxNoOfTables)
Default: 0 (Min: 0, Max: 4294967039)

…
```

  Use this option together with the `--xml` option to obtain output in XML format.

- `--config-binary-file=`*`path-to-file`*

Command-Line Format	`--config-binary-file=path/to/file`
Introduced	8.0.32-ndb-8.0.32
Type	File name
Default Value	

  Gives the path to the management server's cached binary configuration file (`ndb_`*`nodeID`*`_config.bin.`*`seqno`*). This may be a relative or absolute path. If the management server and the `ndb_config` binary used reside on different hosts, you must use an absolute path.

  This example demonstrates combining `--config-binary-file` with other `ndb_config` options to obtain useful output:

```
> ndb_config --config-binary-file=ndb_50_config.bin.1 --diff-default --type=ndbd
config of [DB] node id 5 that is different from default
CONFIG_PARAMETER,ACTUAL_VALUE,DEFAULT_VALUE
```

```
NodeId,5,(mandatory)
BackupDataDir,/home/jon/data/8.0,(null)
DataDir,/home/jon/data/8.0,.
DataMemory,2G,98M
FileSystemPath,/home/jon/data/8.0,(null)
HostName,127.0.0.1,localhost
Nodegroup,0,(null)
ThreadConfig,,(null)

config of [DB] node id 6 that is different from default
CONFIG_PARAMETER,ACTUAL_VALUE,DEFAULT_VALUE
NodeId,6,(mandatory)
BackupDataDir,/home/jon/data/8.0,(null)
DataDir,/home/jon/data/8.0,.
DataMemory,2G,98M
FileSystemPath,/home/jon/data/8.0,(null)
HostName,127.0.0.1,localhost
Nodegroup,0,(null)
ThreadConfig,,(null)

> ndb_config --config-binary-file=ndb_50_config.bin.1 --diff-default --system
config of [SYSTEM] system
CONFIG_PARAMETER,ACTUAL_VALUE,DEFAULT_VALUE
Name,MC_20220216092809,(mandatory)
ConfigGenerationNumber,1,0
PrimaryMGMNode,50,0
```

The relevant portions of the `config.ini` file are shown here:

```
[ndbd default]
DataMemory= 2G
NoOfReplicas= 2

[ndb_mgmd]
NodeId= 50
HostName= 127.0.0.1

[ndbd]
NodeId= 5
HostName= 127.0.0.1
DataDir= /home/jon/data/8.0

[ndbd]
NodeId= 6
HostName= 127.0.0.1
DataDir= /home/jon/data/8.0
```

By comparing the output with the configuration file, you can see that all of the settings in the file have been written by the management server to the binary cache, and thus, applied to the cluster.

- `--config-file=`*`path-to-file`*

Command-Line Format	`--config-file=file_name`
Type	File name
Default Value	

Gives the path to the cluster configuration file (`config.ini`). This may be a relative or absolute path. If the management server and the `ndb_config` binary used reside on different hosts, you must use an absolute path.

- `--config_from_node=#`

Command-Line Format	`--config-from-node=#`
Type	Numeric
Default Value	`none`

Minimum Value	1
Maximum Value	48

Obtain the cluster's configuration data from the data node that has this ID.

If the node having this ID is not a data node, `ndb_config` fails with an error. (To obtain configuration data from the management node instead, simply omit this option.)

- `--connections`

Command-Line Format	`--connections`

Tells `ndb_config` to print `CONNECTIONS` information only—that is, information about parameters found in the `[tcp]`, `[tcp default]`, `[shm]`, or `[shm default]` sections of the cluster configuration file (see Section 23.4.3.10, “NDB Cluster TCP/IP Connections”, and Section 23.4.3.12, “NDB Cluster Shared-Memory Connections”, for more information).

This option is mutually exclusive with `--nodes` and `--system`; only one of these 3 options can be used.

- `--diff-default`

Command-Line Format	`--diff-default`

Print only configuration parameters that have non-default values.

- `--fields=`*`delimiter`*, `-f` *`delimiter`*

Command-Line Format	`--fields=string`
Type	String
Default Value	

Specifies a *`delimiter`* string used to separate the fields in the result. The default is `,` (the comma character).

Note

If the *`delimiter`* contains spaces or escapes (such as `\n` for the linefeed character), then it must be quoted.

- `--host=`*`hostname`*

Command-Line Format	`--host=name`
Type	String
Default Value	

Specifies the host name of the node for which configuration information is to be obtained.

Note

While the hostname `localhost` usually resolves to the IP address `127.0.0.1`, this may not necessarily be true for all operating platforms and configurations. This means that it is possible, when `localhost` is used in `config.ini`, for `ndb_config --host=localhost` to fail if `ndb_config` is run on a different host where `localhost` resolves to a different address (for example, on some versions of SUSE Linux, this is `127.0.0.2`). In general, for best results, you should use numeric IP addresses for all NDB

Cluster configuration values relating to hosts, or verify that all NDB Cluster hosts handle `localhost` in the same fashion.

- `--mycnf`

Command-Line Format	`--mycnf`

Read configuration data from the `my.cnf` file.

- `--ndb-connectstring=connection_string`, `-c connection_string`

Command-Line Format	`--ndb-connectstring=connection_string`
Type	String
Default Value	`[none]`

Specifies the connection string to use in connecting to the management server. The format for the connection string is the same as described in Section 23.4.3.3, “NDB Cluster Connection Strings”, and defaults to `localhost:1186`.

- `--no-defaults`

Command-Line Format	`--no-defaults`

Do not read default options from any option file other than login file.

- `--nodeid=node_id`

Command-Line Format	`--ndb-nodeid=#`
Removed	8.0.31
Type	Integer
Default Value	`[none]`

Specify the node ID of the node for which configuration information is to be obtained.

- `--nodes`

Command-Line Format	`--nodes`

Tells `ndb_config` to print information relating only to parameters defined in an `[ndbd]` or `[ndbd default]` section of the cluster configuration file (see Section 23.4.3.6, “Defining NDB Cluster Data Nodes”).

This option is mutually exclusive with `--connections` and `--system`; only one of these 3 options can be used.

- `--query=query-options`, `-q query-options`

Command-Line Format	`--query=string`
Type	String

Default Value	

This is a comma-delimited list of *query options*—that is, a list of one or more node attributes to be returned. These include `nodeid` (node ID), type (node type—that is, `ndbd`, `mysqld`, or `ndb_mgmd`), and any configuration parameters whose values are to be obtained.

For example, `--query=nodeid,type,datamemory,datadir` returns the node ID, node type, `DataMemory`, and `DataDir` for each node.

Note

If a given parameter is not applicable to a certain type of node, than an empty string is returned for the corresponding value. See the examples later in this section for more information.

- `--query-all`, `-a`

Command-Line Format	`--query-all`
Type	String
Default Value	

Returns a comma-delimited list of all query options (node attributes; note that this list is a single string.

- `--rows=`*`separator`*, `-r` *`separator`*

Command-Line Format	`--rows=string`
Type	String
Default Value	

Specifies a *`separator`* string used to separate the rows in the result. The default is a space character.

Note

If the *`separator`* contains spaces or escapes (such as `\n` for the linefeed character), then it must be quoted.

- `--system`

Command-Line Format	`--system`

Tells `ndb_config` to print `SYSTEM` information only. This consists of system variables that cannot be changed at run time; thus, there is no corresponding section of the cluster configuration file for them. They can be seen (prefixed with `****** SYSTEM ******`) in the output of `ndb_config --configinfo`.

This option is mutually exclusive with `--nodes` and `--connections`; only one of these 3 options can be used.

- `--type=`*`node_type`*

Command-Line Format	`--type=name`
Type	Enumeration
Default Value	`[none]`
Valid Values	`ndbd`

	`mysqld`
	
	`ndb_mgmd`

Filters results so that only configuration values applying to nodes of the specified *`node_type`* (`ndbd`, `mysqld`, or `ndb_mgmd`) are returned.

- `--usage`, `--help`, or `-?`

Command-Line Format	`--help`

Causes `ndb_config` to print a list of available options, and then exit.

- `--version`, `-V`

Command-Line Format	`--version`

Causes `ndb_config` to print a version information string, and then exit.

- `--configinfo --xml`

Command-Line Format	`--configinfo --xml`

Cause `ndb_config --configinfo` to provide output as XML by adding this option. A portion of such output is shown in this example:

```
$> ndb_config --configinfo --xml

<configvariables protocolversion="1" ndbversionstring="5.7.42-ndb-7.5.31"
                    ndbversion="460032" ndbversionmajor="7" ndbversionminor="5"
                    ndbversionbuild="0">
  <section name="SYSTEM">
    <param name="Name" comment="Name of system (NDB Cluster)" type="string"
               mandatory="true"/>
    <param name="PrimaryMGMNode" comment="Node id of Primary ndb_mgmd(MGM) node"
               type="unsigned" default="0" min="0" max="4294967039"/>
    <param name="ConfigGenerationNumber" comment="Configuration generation number"
               type="unsigned" default="0" min="0" max="4294967039"/>
  </section>
  <section name="MYSQLD" primarykeys="NodeId">
    <param name="wan" comment="Use WAN TCP setting as default" type="bool"
               default="false"/>
    <param name="HostName" comment="Name of computer for this node"
               type="string" default=""/>
    <param name="Id" comment="NodeId" type="unsigned" mandatory="true"
               min="1" max="255" deprecated="true"/>
    <param name="NodeId" comment="Number identifying application node (mysqld(API))"
               type="unsigned" mandatory="true" min="1" max="255"/>
    <param name="ExecuteOnComputer" comment="HostName" type="string"
               deprecated="true"/>

    …

  </section>

  …

</configvariables>
```

Note

Normally, the XML output produced by `ndb_config --configinfo --xml` is formatted using one line per element; we have added extra whitespace in the previous example, as well as the next one, for reasons of legibility. This should not make any difference to applications using this output, since most

> XML processors either ignore nonessential whitespace as a matter of course, or can be instructed to do so.

The XML output also indicates when changing a given parameter requires that data nodes be restarted using the `--initial` option. This is shown by the presence of an `initial="true"` attribute in the corresponding `<param>` element. In addition, the restart type (`system` or `node`) is also shown; if a given parameter requires a system restart, this is indicated by the presence of a `restart="system"` attribute in the corresponding `<param>` element. For example, changing the value set for the `Diskless` parameter requires a system initial restart, as shown here (with the `restart` and `initial` attributes highlighted for visibility):

```
<param name="Diskless" comment="Run wo/ disk" type="bool" default="false"
          restart="system" initial="true"/>
```

Currently, no `initial` attribute is included in the XML output for `<param>` elements corresponding to parameters which do not require initial restarts; in other words, `initial="false"` is the default, and the value `false` should be assumed if the attribute is not present. Similarly, the default restart type is `node` (that is, an online or “rolling” restart of the cluster), but the `restart` attribute is included only if the restart type is `system` (meaning that all cluster nodes must be shut down at the same time, then restarted).

Deprecated parameters are indicated in the XML output by the `deprecated` attribute, as shown here:

```
<param name="NoOfDiskPagesToDiskAfterRestartACC" comment="DiskCheckpointSpeed"
       type="unsigned" default="20" min="1" max="4294967039" deprecated="true"/>
```

In such cases, the `comment` refers to one or more parameters that supersede the deprecated parameter. Similarly to `initial`, the `deprecated` attribute is indicated only when the parameter is deprecated, with `deprecated="true"`, and does not appear at all for parameters which are not deprecated. (Bug #21127135)

Parameters that are required are indicated with `mandatory="true"`, as shown here:

```
<param name="NodeId"
          comment="Number identifying application node (mysqld(API))"
          type="unsigned" mandatory="true" min="1" max="255"/>
```

In much the same way that the `initial` or `deprecated` attribute is displayed only for a parameter that requires an initial restart or that is deprecated, the `mandatory` attribute is included only if the given parameter is actually required.

> **Important**
>
> The `--xml` option can be used only with the `--configinfo` option. Using `--xml` without `--configinfo` fails with an error.

Unlike the options used with this program to obtain current configuration data, `--configinfo` and `--xml` use information obtained from the NDB Cluster sources when `ndb_config` was compiled. For this reason, no connection to a running NDB Cluster or access to a `config.ini` or `my.cnf` file is required for these two options.

- `--print-defaults`

Command-Line Format	`--print-defaults`

Print program argument list and exit.

- `--defaults-file`

Command-Line Format	`--defaults-file=path`

Type	String
Default Value	[none]

Read default options from given file only.

- --defaults-extra-file

Command-Line Format	--defaults-extra-file=path
Type	String
Default Value	[none]

Read given file after global files are read.

- --defaults-group-suffix

Command-Line Format	--defaults-group-suffix=string
Type	String
Default Value	[none]

Also read groups with concat(group, suffix).

- --login-path

Command-Line Format	--login-path=path
Type	String
Default Value	[none]

Read given path from login file.

- --help

Command-Line Format	--help

Display help text and exit.

- --connect-string

Command-Line Format	--connect-string=connection_string
Type	String
Default Value	[none]

Same as --ndb-connectstring.

- --ndb-mgmd-host

Command-Line Format	--ndb-mgmd-host=connection_string
Type	String
Default Value	[none]

Same as --ndb-connectstring.

- --ndb-nodeid

Command-Line Format	--ndb-nodeid=#
Removed	8.0.31

Type	Integer
Default Value	[none]

Set node ID for this node, overriding any ID set by --ndb-connectstring.

- --core-file

Command-Line Format	--core-file
Removed	8.0.31

Write core file on error; used in debugging.

- --character-sets-dir

Command-Line Format	--character-sets-dir=path
Removed	8.0.31

Directory containing character sets.

- --connect-retries

Command-Line Format	--connect-retries=#
Type	Integer
Default Value	12
Minimum Value	0
Maximum Value	12

Number of times to retry connection before giving up.

- --connect-retry-delay

Command-Line Format	--connect-retry-delay=#
Type	Integer
Default Value	5
Minimum Value	0
Maximum Value	5

Number of seconds to wait between attempts to contact management server.

- --ndb-optimized-node-selection

Command-Line Format	--ndb-optimized-node-selection
Removed	8.0.31

Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable.

Combining other ndb_config options (such as --query or --type) with --configinfo (with or without the --xml option is not supported. Currently, if you attempt to do so, the usual result is that all other options besides --configinfo or --xml are simply ignored. *However, this behavior is not guaranteed and is subject to change at any time.* In addition, since ndb_config, when used with the --configinfo option, does not access the NDB Cluster or read any files, trying to specify additional options such as --ndb-connectstring or --config-file with --configinfo serves no purpose.

Examples

1. To obtain the node ID and type of each node in the cluster:

```
$> ./ndb_config --query=nodeid,type --fields=':' --rows='\n'
1:ndbd
2:ndbd
3:ndbd
4:ndbd
5:ndb_mgmd
6:mysqld
7:mysqld
8:mysqld
9:mysqld
```

   In this example, we used the `--fields` options to separate the ID and type of each node with a colon character (`:`), and the `--rows` options to place the values for each node on a new line in the output.

2. To produce a connection string that can be used by data, SQL, and API nodes to connect to the management server:

```
$> ./ndb_config --config-file=usr/local/mysql/cluster-data/config.ini \
--query=hostname,portnumber --fields=: --rows=, --type=ndb_mgmd
198.51.100.179:1186
```

3. This invocation of `ndb_config` checks only data nodes (using the `--type` option), and shows the values for each node's ID and host name, as well as the values set for its `DataMemory` and `DataDir` parameters:

```
$> ./ndb_config --type=ndbd --query=nodeid,host,datamemory,datadir -f ' : ' -r '\n'
1 : 198.51.100.193 : 83886080 : /usr/local/mysql/cluster-data
2 : 198.51.100.112 : 83886080 : /usr/local/mysql/cluster-data
3 : 198.51.100.176 : 83886080 : /usr/local/mysql/cluster-data
4 : 198.51.100.119 : 83886080 : /usr/local/mysql/cluster-data
```

   In this example, we used the short options `-f` and `-r` for setting the field delimiter and row separator, respectively, as well as the short option `-q` to pass a list of parameters to be obtained.

4. To exclude results from any host except one in particular, use the `--host` option:

```
$> ./ndb_config --host=198.51.100.176 -f : -r '\n' -q id,type
3:ndbd
5:ndb_mgmd
```

   In this example, we also used the short form `-q` to determine the attributes to be queried.

   Similarly, you can limit results to a node with a specific ID using the `--nodeid` option.

23.5.8 ndb_delete_all — Delete All Rows from an NDB Table

`ndb_delete_all` deletes all rows from the given `NDB` table. In some cases, this can be much faster than `DELETE` or even `TRUNCATE TABLE`.

Usage

```
ndb_delete_all -c connection_string tbl_name -d db_name
```

This deletes all rows from the table named *`tbl_name`* in the database named *`db_name`*. It is exactly equivalent to executing `TRUNCATE db_name.tbl_name` in MySQL.

Options that can be used with `ndb_delete_all` are shown in the following table. Additional descriptions follow the table.

Table 23.30 Command-line options used with the program ndb_delete_all

Format	Description	Added, Deprecated, or Removed
`--character-sets-dir=path`	Directory containing character sets	REMOVED: 8.0.31
`--connect-retries=#`	Number of times to retry connection before giving up	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retry-delay=#`	Number of seconds to wait between attempts to contact management server	(Supported in all NDB releases based on MySQL 8.0)
`--connect-string=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--core-file`	Write core file on error; used in debugging	REMOVED: 8.0.31
`--database=name`, `-d name`	Name of the database in which the table is found	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-extra-file=path`	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-file=path`	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-group-suffix=string`	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)
`--diskscan`	Perform disk scan	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-connectstring=connection_string`, `-c connection_string`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-mgmd-host=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable	REMOVED: 8.0.31

Format	Description	Added, Deprecated, or Removed
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)
`--transactional`, `-t`	Perform delete in one single transaction; possible to run out of operations when used	(Supported in all NDB releases based on MySQL 8.0)
`--tupscan`	Perform tuple scan	(Supported in all NDB releases based on MySQL 8.0)
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)

- `--character-sets-dir`

Command-Line Format	`--character-sets-dir=path`
Removed	8.0.31

Directory containing character sets.

- `--connect-retries`

Command-Line Format	`--connect-retries=#`
Type	Integer
Default Value	`12`
Minimum Value	`0`
Maximum Value	`12`

Number of times to retry connection before giving up.

- `--connect-retry-delay`

Command-Line Format	`--connect-retry-delay=#`
Type	Integer
Default Value	`5`
Minimum Value	`0`
Maximum Value	`5`

Number of seconds to wait between attempts to contact management server.

- `--connect-string`

Command-Line Format	`--connect-string=connection_string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--core-file`

Command-Line Format	`--core-file`
Removed	8.0.31

Write core file on error; used in debugging.

- `--database`, `-d`

Command-Line Format	`--database=name`
Type	String
Default Value	`TEST_DB`

Name of the database containing the table to delete from.

- `--defaults-extra-file`

Command-Line Format	`--defaults-extra-file=path`
Type	String
Default Value	`[none]`

Read given file after global files are read.

- `--defaults-file`

Command-Line Format	`--defaults-file=path`
Type	String
Default Value	`[none]`

Read default options from given file only.

- `--defaults-group-suffix`

Command-Line Format	`--defaults-group-suffix=string`
Type	String
Default Value	`[none]`

Also read groups with concat(group, suffix).

- `--diskscan`

Command-Line Format	`--diskscan`

Run a disk scan.

- `--help`

Command-Line Format	`--help`

Display help text and exit.

- `--login-path`

Command-Line Format	`--login-path=path`

Default Value	[none]

Read given path from login file.

- `--ndb-connectstring`

Command-Line Format	`--ndb-connectstring=connection_string`
Type	String
Default Value	[none]

Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;][host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf.

- `--ndb-mgmd-host`

Command-Line Format	`--ndb-mgmd-host=connection_string`
Type	String
Default Value	[none]

Same as `--ndb-connectstring`.

- `--ndb-nodeid`

Command-Line Format	`--ndb-nodeid=#`
Type	Integer
Default Value	[none]

Set node ID for this node, overriding any ID set by `--ndb-connectstring`.

- `--ndb-optimized-node-selection`

Command-Line Format	`--ndb-optimized-node-selection`
Removed	8.0.31

Enable optimizations for selection of nodes for transactions. Enabled by default; use `--skip-ndb-optimized-node-selection` to disable.

- `--no-defaults`

Command-Line Format	`--no-defaults`

Do not read default options from any option file other than login file.

- `--print-defaults`

Command-Line Format	`--print-defaults`

Print program argument list and exit.

- `--transactional`, `-t`

  Use of this option causes the delete operation to be performed as a single transaction.

  

  Warning

  With very large tables, using this option may cause the number of operations available to the cluster to be exceeded.

- `--tupscan`

  Run a tuple scan.

- `--usage`

Command-Line Format	`--usage`

  Display help text and exit; same as `--help`.

- `--version`

Command-Line Format	`--version`

  Display version information and exit.

In NDB 7.6 and earlier, this program printed `NDBT_ProgramExit - status` upon completion of its run, due to an unnecessary dependency on the `NDBT` testing library. This dependency has been removed in NDB 8.0, eliminating the extraneous output.

23.5.9 ndb_desc — Describe NDB Tables

`ndb_desc` provides a detailed description of one or more `NDB` tables.

Usage

```
ndb_desc -c connection_string tbl_name -d db_name [options]

ndb_desc -c connection_string index_name -d db_name -t tbl_name
```

Additional options that can be used with `ndb_desc` are listed later in this section.

Sample Output

MySQL table creation and population statements:

```
USE test;

CREATE TABLE fish (
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    length_mm INT NOT NULL,
    weight_gm INT NOT NULL,

    PRIMARY KEY pk (id),
    UNIQUE KEY uk (name)
) ENGINE=NDB;

INSERT INTO fish VALUES
    (NULL, 'guppy', 35, 2), (NULL, 'tuna', 2500, 150000),
    (NULL, 'shark', 3000, 110000), (NULL, 'manta ray', 1500, 50000),
    (NULL, 'grouper', 900, 125000), (NULL ,'puffer', 250, 2500);
```

Output from `ndb_desc`:

```
$> ./ndb_desc -c localhost fish -d test -p
-- fish --
```

```
Version: 2
Fragment type: HashMapPartition
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 4
Number of primary keys: 1
Length of frm data: 337
Max Rows: 0
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
PartitionCount: 2
FragmentCount: 2
PartitionBalance: FOR_RP_BY_LDM
ExtraRowGciBits: 0
ExtraRowAuthorBits: 0
TableStatus: Retrieved
Table options:
HashMap: DEFAULT-HASHMAP-3840-2
-- Attributes --
id Int PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY DYNAMIC
length_mm Int NOT NULL AT=FIXED ST=MEMORY DYNAMIC
weight_gm Int NOT NULL AT=FIXED ST=MEMORY DYNAMIC
-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
uk(name) - OrderedIndex
uk$unique(name) - UniqueHashIndex
-- Per partition info --
Partition       Row count       Commit count    Frag fixed memory       Frag varsized memory    Extent_
0               2               2               32768                   32768                   0
1               4               4               32768                   32768                   0


NDBT_ProgramExit: 0 - OK
```

Information about multiple tables can be obtained in a single invocation of `ndb_desc` by using their names, separated by spaces. All of the tables must be in the same database.

You can obtain additional information about a specific index using the `--table` (short form: `-t`) option and supplying the name of the index as the first argument to `ndb_desc`, as shown here:

```
$> ./ndb_desc uk -d test -t fish
-- uk --
Version: 2
Base table: fish
Number of attributes: 1
Logging: 0
Index type: OrderedIndex
Index status: Retrieved
-- Attributes --
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
-- IndexTable 10/uk --
Version: 2
Fragment type: FragUndefined
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: yes
Number of attributes: 2
Number of primary keys: 1
Length of frm data: 0
Max Rows: 0
Row Checksum: 1
Row GCI: 1
SingleUserMode: 2
ForceVarPart: 0
PartitionCount: 2
```

```
FragmentCount: 2
FragmentCountType: ONE_PER_LDM_PER_NODE
ExtraRowGciBits: 0
ExtraRowAuthorBits: 0
TableStatus: Retrieved
Table options:
-- Attributes --
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
NDB$TNODE Unsigned [64] PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY
-- Indexes --
PRIMARY KEY(NDB$TNODE) - UniqueHashIndex

NDBT_ProgramExit: 0 - OK
```

When an index is specified in this way, the `--extra-partition-info` and `--extra-node-info` options have no effect.

The `Version` column in the output contains the table's schema object version. For information about interpreting this value, see NDB Schema Object Versions.

Three of the table properties that can be set using `NDB_TABLE` comments embedded in `CREATE TABLE` and `ALTER TABLE` statements are also visible in `ndb_desc` output. The table's `FRAGMENT_COUNT_TYPE` is always shown in the `FragmentCountType` column. `READ_ONLY` and `FULLY_REPLICATED`, if set to 1, are shown in the `Table options` column. You can see this after executing the following `ALTER TABLE` statement in the `mysql` client:

```
mysql> ALTER TABLE fish COMMENT='NDB_TABLE=READ_ONLY=1,FULLY_REPLICATED=1';
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
+---------+------+-----------------------------------------------------------------------------------------------
| Level   | Code | Message
+---------+------+-----------------------------------------------------------------------------------------------
| Warning | 1296 | Got error 4503 'Table property is FRAGMENT_COUNT_TYPE=ONE_PER_LDM_PER_NODE but not in co
+---------+------+-----------------------------------------------------------------------------------------------
1 row in set (0.00 sec)
```

The warning is issued because `READ_ONLY=1` requires that the table's fragment count type is (or be set to) `ONE_PER_LDM_PER_NODE_GROUP`; `NDB` sets this automatically in such cases. You can check that the `ALTER TABLE` statement has the desired effect using `SHOW CREATE TABLE`:

```
mysql> SHOW CREATE TABLE fish\G
*************************** 1. row ***************************
       Table: fish
Create Table: CREATE TABLE `fish` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  `length_mm` int(11) NOT NULL,
  `weight_gm` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk` (`name`)
) ENGINE=ndbcluster DEFAULT CHARSET=latin1
COMMENT='NDB_TABLE=READ_BACKUP=1,FULLY_REPLICATED=1'
1 row in set (0.01 sec)
```

Because `FRAGMENT_COUNT_TYPE` was not set explicitly, its value is not shown in the comment text printed by `SHOW CREATE TABLE`. `ndb_desc`, however, displays the updated value for this attribute. The `Table options` column shows the binary properties just enabled. You can see this in the output shown here (emphasized text):

```
$> ./ndb_desc -c localhost fish -d test -p
-- fish --
Version: 4
Fragment type: HashMapPartition
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 4
```

```
Number of primary keys: 1
Length of frm data: 380
Max Rows: 0
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
PartitionCount: 1
FragmentCount: 1
FragmentCountType: ONE_PER_LDM_PER_NODE_GROUP
ExtraRowGciBits: 0
ExtraRowAuthorBits: 0
TableStatus: Retrieved
Table options: readbackup, fullyreplicated
HashMap: DEFAULT-HASHMAP-3840-1
-- Attributes --
id Int PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
name Varchar(20;latin1_swedish_ci) NOT NULL AT=SHORT_VAR ST=MEMORY DYNAMIC
length_mm Int NOT NULL AT=FIXED ST=MEMORY DYNAMIC
weight_gm Int NOT NULL AT=FIXED ST=MEMORY DYNAMIC
-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
uk(name) - OrderedIndex
uk$unique(name) - UniqueHashIndex
-- Per partition info --
Partition       Row count       Commit count    Frag fixed memory       Frag varsized memory    Extent_

NDBT_ProgramExit: 0 - OK
```

For more information about these table properties, see Section 13.1.20.12, “Setting NDB Comment Options”.

The `Extent_space` and `Free extent_space` columns are applicable only to `NDB` tables having columns on disk; for tables having only in-memory columns, these columns always contain the value `0`.

To illustrate their use, we modify the previous example. First, we must create the necessary Disk Data objects, as shown here:

```
CREATE LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_1.log'
    INITIAL_SIZE 16M
    UNDO_BUFFER_SIZE 2M
    ENGINE NDB;

ALTER LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_2.log'
    INITIAL_SIZE 12M
    ENGINE NDB;

CREATE TABLESPACE ts_1
    ADD DATAFILE 'data_1.dat'
    USE LOGFILE GROUP lg_1
    INITIAL_SIZE 32M
    ENGINE NDB;

ALTER TABLESPACE ts_1
    ADD DATAFILE 'data_2.dat'
    INITIAL_SIZE 48M
    ENGINE NDB;
```

(For more information on the statements just shown and the objects created by them, see Section 23.6.11.1, “NDB Cluster Disk Data Objects”, as well as Section 13.1.16, “CREATE LOGFILE GROUP Statement”, and Section 13.1.21, “CREATE TABLESPACE Statement”.)

Now we can create and populate a version of the `fish` table that stores 2 of its columns on disk (deleting the previous version of the table first, if it already exists):

```
DROP TABLE IF EXISTS fish;
```

```
CREATE TABLE fish (
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    length_mm INT NOT NULL,
    weight_gm INT NOT NULL,

    PRIMARY KEY pk (id),
    UNIQUE KEY uk (name)
) TABLESPACE ts_1 STORAGE DISK
ENGINE=NDB;

INSERT INTO fish VALUES
    (NULL, 'guppy', 35, 2), (NULL, 'tuna', 2500, 150000),
    (NULL, 'shark', 3000, 110000), (NULL, 'manta ray', 1500, 50000),
    (NULL, 'grouper', 900, 125000), (NULL ,'puffer', 250, 2500);
```

When run against this version of the table, `ndb_desc` displays the following output:

```
$> ./ndb_desc -c localhost fish -d test -p
-- fish --
Version: 1
Fragment type: HashMapPartition
K Value: 6
Min load factor: 78
Max load factor: 80
Temporary table: no
Number of attributes: 4
Number of primary keys: 1
Length of frm data: 1001
Max Rows: 0
Row Checksum: 1
Row GCI: 1
SingleUserMode: 0
ForceVarPart: 1
PartitionCount: 2
FragmentCount: 2
PartitionBalance: FOR_RP_BY_LDM
ExtraRowGciBits: 0
ExtraRowAuthorBits: 0
TableStatus: Retrieved
Table options: readbackup
HashMap: DEFAULT-HASHMAP-3840-2
Tablespace id: 16
Tablespace: ts_1
-- Attributes --
id Int PRIMARY KEY DISTRIBUTION KEY AT=FIXED ST=MEMORY AUTO_INCR
name Varchar(80;utf8mb4_0900_ai_ci) NOT NULL AT=SHORT_VAR ST=MEMORY
length_mm Int NOT NULL AT=FIXED ST=DISK
weight_gm Int NOT NULL AT=FIXED ST=DISK
-- Indexes --
PRIMARY KEY(id) - UniqueHashIndex
PRIMARY(id) - OrderedIndex
uk(name) - OrderedIndex
uk$unique(name) - UniqueHashIndex
-- Per partition info --
Partition       Row count       Commit count    Frag fixed memory       Frag varsized memory    Extent_spac
0               2               2               32768                   32768                   1048576
1               4               4               32768                   32768                   1048576


NDBT_ProgramExit: 0 - OK
```

This means that 1048576 bytes are allocated from the tablespace for this table on each partition, of which 1044440 bytes remain free for additional storage. In other words, 1048576 - 1044440 = 4136 bytes per partition is currently being used to store the data from this table's disk-based columns. The number of bytes shown as `Free extent_space` is available for storing on-disk column data from the `fish` table only; for this reason, it is not visible when selecting from the Information Schema `FILES` table.

`Tablespace id` and `Tablespace` are displayed for Disk Data tables beginning with NDB 8.0.21.

For fully replicated tables, `ndb_desc` shows only the nodes holding primary partition fragment replicas; nodes with copy fragment replicas (only) are ignored. You can obtain such information, using the `mysql` client, from the `table_distribution_status`, `table_fragments`, `table_info`, and `table_replicas` tables in the `ndbinfo` database.

All options that can be used with `ndb_desc` are shown in the following table. Additional descriptions follow the table.

Table 23.31 Command-line options used with the program ndb_desc

Format	Description	Added, Deprecated, or Removed
`--auto-inc`, `-a`	Show next value for AUTO_INCREMENT oolumn if table has one	ADDED: NDB 8.0.21
`--blob-info`, `-b`	Include partition information for BLOB tables in output. Requires that the -p option also be used	(Supported in all NDB releases based on MySQL 8.0)
`--character-sets-dir=path`	Directory containing character sets	REMOVED: 8.0.31
`--connect-retries=#`	Number of times to retry connection before giving up	(Supported in all NDB releases based on MySQL 8.0)
`--connect-retry-delay=#`	Number of seconds to wait between attempts to contact management server	(Supported in all NDB releases based on MySQL 8.0)
`--connect-string=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--context`, `-x`	Show extra information for table such as database, schema, name, and internal ID	ADDED: NDB 8.0.21
`--core-file`	Write core file on error; used in debugging	REMOVED: 8.0.31
`--database=name`, `-d name`	Name of database containing table	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-extra-file=path`	Read given file after global files are read	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-file=path`	Read default options from given file only	(Supported in all NDB releases based on MySQL 8.0)
`--defaults-group-suffix=string`	Also read groups with concat(group, suffix)	(Supported in all NDB releases based on MySQL 8.0)
`--extra-node-info`, `-n`	Include partition-to-data-node mappings in output; requires --extra-partition-info	(Supported in all NDB releases based on MySQL 8.0)
`--extra-partition-info`, `-p`	Display information about partitions	(Supported in all NDB releases based on MySQL 8.0)
`--help`, `-?`	Display help text and exit	(Supported in all NDB releases based on MySQL 8.0)
`--login-path=path`	Read given path from login file	(Supported in all NDB releases based on MySQL 8.0)

Format	Description	Added, Deprecated, or Removed
`--ndb-connectstring=connection_string`, `-c connection_string`	Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;] [host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-mgmd-host=connection_string`, `-c connection_string`	Same as --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-nodeid=#`	Set node ID for this node, overriding any ID set by --ndb-connectstring	(Supported in all NDB releases based on MySQL 8.0)
`--ndb-optimized-node-selection`	Enable optimizations for selection of nodes for transactions. Enabled by default; use --skip-ndb-optimized-node-selection to disable	REMOVED: 8.0.31
`--no-defaults`	Do not read default options from any option file other than login file	(Supported in all NDB releases based on MySQL 8.0)
`--print-defaults`	Print program argument list and exit	(Supported in all NDB releases based on MySQL 8.0)
`--retries=#`, `-r #`	Number of times to retry the connection (once per second)	(Supported in all NDB releases based on MySQL 8.0)
`--table=name`, `-t name`	Specify the table in which to find an index. When this option is used, -p and -n have no effect and are ignored	(Supported in all NDB releases based on MySQL 8.0)
`--unqualified`, `-u`	Use unqualified table names	(Supported in all NDB releases based on MySQL 8.0)
`--usage`, `-?`	Display help text and exit; same as --help	(Supported in all NDB releases based on MySQL 8.0)
`--version`, `-V`	Display version information and exit	(Supported in all NDB releases based on MySQL 8.0)

- `--auto-inc`, `-a`

  Show the next value for a table's `AUTO_INCREMENT` column, if it has one.

- `--blob-info`, `-b`

  Include information about subordinate `BLOB` and `TEXT` columns.

  Use of this option also requires the use of the `--extra-partition-info` (`-p`) option.

- `--character-sets-dir`

Command-Line Format	`--character-sets-dir=path`

Removed	8.0.31

Directory containing character sets.

- `--connect-retries`

Command-Line Format	`--connect-retries=#`
Type	Integer
Default Value	`12`
Minimum Value	`0`
Maximum Value	`12`

Number of times to retry connection before giving up.

- `--connect-retry-delay`

Command-Line Format	`--connect-retry-delay=#`
Type	Integer
Default Value	`5`
Minimum Value	`0`
Maximum Value	`5`

Number of seconds to wait between attempts to contact management server.

- `--connect-string`

Command-Line Format	`--connect-string=connection_string`
Type	String
Default Value	`[none]`

Same as `--ndb-connectstring`.

- `--context`, `-x`

Show additional contextual information for the table such as schema, database name, table name, and the table's internal ID.

- `--core-file`

Command-Line Format	`--core-file`
Removed	8.0.31

Write core file on error; used in debugging.

- `--database=`*`db_name`*, `-d`

Specify the database in which the table should be found.

- `--defaults-extra-file`

Command-Line Format	`--defaults-extra-file=path`
Type	String
Default Value	`[none]`

Read given file after global files are read.

- `--defaults-file`

Command-Line Format	`--defaults-file=path`
Type	String
Default Value	`[none]`

Read default options from given file only.

- `--defaults-group-suffix`

Command-Line Format	`--defaults-group-suffix=string`
Type	String
Default Value	`[none]`

Also read groups with concat(group, suffix).

- `--extra-node-info`, `-n`

Include information about the mappings between table partitions and the data nodes upon which they reside. This information can be useful for verifying distribution awareness mechanisms and supporting more efficient application access to the data stored in NDB Cluster.

Use of this option also requires the use of the `--extra-partition-info` (`-p`) option.

- `--extra-partition-info`, `-p`

Print additional information about the table's partitions.

- `--help`

Command-Line Format	`--help`

Display help text and exit.

- `--login-path`

Command-Line Format	`--login-path=path`
Type	String
Default Value	`[none]`

Read given path from login file.

- `--ndb-connectstring`

Command-Line Format	`--ndb-connectstring=connection_string`
Type	String
Default Value	`[none]`

Set connect string for connecting to ndb_mgmd. Syntax: "[nodeid=id;][host=]hostname[:port]". Overrides entries in NDB_CONNECTSTRING and my.cnf.

- `--ndb-mgmd-host`

Command-Line Format	`--ndb-mgmd-host=connection_string`
Type	String