

MySQL™ Cluster Manager 1.3.6 User Manual

Abstract

Deprecated release. MySQL Cluster Manager 1.3 has been superseded by later releases that add new features and incorporate fixes to bugs found in previous MySQL Cluster Manager releases; users of MySQL Cluster Manager 1.3.6 should upgrade as soon as possible. Thus, this document is archival in nature, and no longer actively maintained. Information about the most recent MySQL Cluster Manager release can be found at <http://dev.mysql.com/doc/mysql-cluster-manager/en/>.

This is the User Manual for the *MySQL™ Cluster Manager*, version 1.3.6. It documents the *MySQL Cluster Manager Agent* and *MySQL Cluster Manager Client* software applications which can be used to administer *MySQL Cluster*, a version of the *MySQL Database System* (referred to hereafter as “MySQL Server” or simply “MySQL”) that incorporates the [NDB](#) storage engine for high availability and data redundancy in a distributed computing environment.

This Manual applies to MySQL Cluster Manager 1.3.6 and contains information that may not apply to older versions of the MySQL Cluster Manager software. For documentation covering previous MySQL Cluster Manager releases, see [MySQL Documentation: MySQL Cluster](#), on the MySQL website.

MySQL Cluster Manager features. This manual describes features that may not be included in every version of MySQL Cluster Manager, and such features may not be included in the version of MySQL Cluster Manager licensed to you. If you have any questions about the features included in your version of MySQL Cluster Manager, refer to your MySQL Cluster Manager license agreement or contact your Oracle sales representative.

MySQL Cluster Manager, MySQL Server, and MySQL Cluster features. This manual contains certain basic information about MySQL Server and MySQL Cluster; however, it is not in any way intended as an exhaustive reference for either of these products.

Current versions of MySQL Cluster compatible with MySQL Cluster Manager are based on versions 7.2, 7.3, and 7.4 of the [NDB](#) storage engine; these versions of MySQL Cluster are known as “MySQL Cluster NDB 7.2,” “MySQL Cluster NDB 7.3,” and “MySQL Cluster NDB 7.4,” respectively. MySQL Cluster functionality varies between MySQL Cluster releases; MySQL Cluster Manager cannot supply or emulate MySQL Cluster features that are not present in the version of the MySQL Cluster software in use.

For complete information about MySQL Server and MySQL Cluster, please refer to the appropriate version of the *MySQL Manual*: For information about MySQL Cluster NDB 7.3 and 7.4, see [MySQL NDB Cluster 7.3 and NDB Cluster 7.4](#).

If you do not have the MySQL Server and MySQL Cluster documentation, you can obtain it free of charge from the [MySQL Documentation Library](#), on the MySQL website.

For legal information, see the [Legal Notices](#).

For help with using MySQL, please visit the [MySQL Forums](#), where you can discuss your issues with other MySQL users.

Licensing information. This product may include third-party software, used under license. See the [MySQL Cluster Manager 1.3 License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this release.

Document generated on: 2021-01-29 (revision: 68621)

Table of Contents

Preface and Legal Notices	v
1 Overview of MySQL Cluster Manager	1
1.1 MySQL Cluster Manager Terminology	1
1.2 MySQL Cluster Manager Architecture	2
2 MySQL Cluster Manager Installation, Configuration, Cluster Setup	5
2.1 Obtaining MySQL Cluster Manager	5
2.2 Supported Platforms and MySQL Cluster Versions	5
2.3 MySQL Cluster Manager Installation	6
2.3.1 Installing MySQL Cluster Manager on Unix Platforms	6
2.3.2 Installing MySQL Cluster Manager on Windows Platforms	9
2.3.3 Setting the MySQL Cluster Manager Agent User Name and Password	18
2.4 MySQL Cluster Manager Configuration File	19
2.5 Upgrading MySQL Cluster Manager	20
3 Using MySQL Cluster Manager	23
3.1 <code>mcmd</code> , the MySQL Cluster Manager Agent	23
3.2 Starting and Stopping the MySQL Cluster Manager Agent	31
3.2.1 Starting and Stopping the Agent on Linux	32
3.2.2 Starting and Stopping the MySQL Cluster Manager Agent on Windows	33
3.3 Starting the MySQL Cluster Manager Client	35
3.4 Setting Up MySQL Clusters with MySQL Cluster Manager	38
3.4.1 Creating a MySQL Cluster with MySQL Cluster Manager	38
3.5 Importing MySQL Clusters into MySQL Cluster Manager	39
3.5.1 Importing a Cluster Into MySQL Cluster Manager: Basic Procedure	40
3.5.2 Importing a Cluster Into MySQL Cluster Manager: Example	41
3.6 MySQL Cluster Backup and Restore Using MySQL Cluster Manager	56
3.6.1 Requirements for Backup and Restore	56
3.6.2 Basic MySQL Cluster Backup and Restore Using MySQL Cluster Manager	56
3.7 Backing Up and Restoring MySQL Cluster Manager Agents	60
3.8 Setting Up MySQL Cluster Replication with MySQL Cluster Manager	62
4 MySQL Cluster Manager Client Commands	65
4.1 Online Help and Information Commands	71
4.2 MySQL Cluster Manager Site and Agent Commands	78
4.2.1 The <code>add hosts</code> Command	78
4.2.2 The <code>remove hosts</code> Command	79
4.2.3 The <code>change log-level</code> Command	79
4.2.4 The <code>rotate log</code> Command	80
4.2.5 The <code>collect logs</code> Command	81
4.2.6 The <code>create site</code> Command	81
4.2.7 The <code>delete site</code> Command	82
4.2.8 The <code>list sites</code> Command	83
4.2.9 The <code>list hosts</code> Command	83
4.2.10 The <code>show settings</code> Command	84
4.2.11 The <code>stop agents</code> Command	84
4.2.12 The <code>version</code> Command	85
4.2.13 The <code>show warnings</code> Command	85
4.3 MySQL Cluster Manager Package Commands	86
4.3.1 The <code>add package</code> Command	86
4.3.2 The <code>delete package</code> Command	88
4.3.3 The <code>list packages</code> Command	89
4.4 MySQL Cluster Manager Cluster Commands	90
4.4.1 The <code>create cluster</code> Command	90

4.4.2	The <code>delete cluster</code> Command	94
4.4.3	The <code>list clusters</code> Command	95
4.4.4	The <code>list nextnodeids</code> Command	95
4.4.5	The <code>restart cluster</code> Command	95
4.4.6	The <code>show status</code> Command	96
4.4.7	The <code>start cluster</code> Command	100
4.4.8	The <code>stop cluster</code> Command	101
4.4.9	The <code>upgrade cluster</code> Command	102
4.5	MySQL Cluster Manager Configuration Commands	103
4.5.1	The <code>get</code> Command	106
4.5.2	The <code>reset</code> Command	115
4.5.3	The <code>set</code> Command	122
4.6	MySQL Cluster Manager Process Commands	128
4.6.1	The <code>add process</code> Command	128
4.6.2	The <code>change process</code> Command	131
4.6.3	The <code>list processes</code> Command	134
4.6.4	The <code>start process</code> Command	134
4.6.5	The <code>stop process</code> Command	135
4.6.6	The <code>remove process</code> Command	135
4.7	MySQL Cluster Manager Backup and Restore Commands	137
4.7.1	The <code>abort backup</code> Command	137
4.7.2	The <code>backup cluster</code> Command	137
4.7.3	The <code>list backups</code> Command	138
4.7.4	The <code>restore cluster</code> Command	138
4.7.5	The <code>backup agents</code> Command	140
4.8	MySQL Cluster Manager Cluster Importation Commands	141
4.8.1	The <code>import cluster</code> Command	141
4.8.2	The <code>import config</code> Command	141
5	MySQL Cluster Manager Limitations and Known Issues	143
5.1	MySQL Cluster Manager Usage and Design Limitations	143
5.2	MySQL Cluster Manager 1.3.6 Limitations Relating to the MySQL Server	144
5.3	MySQL Cluster Manager Limitations Relating to MySQL Cluster	144
5.4	Syntax and Related Issues in MySQL Cluster Manager	145
A	Changes in MySQL Cluster Manager	147
B	Attribute Summary Tables	149
B.1	Management Node Configuration Parameters	149
B.2	Data Node Configuration Parameters	150
B.3	API Node Configuration Parameters	156
B.4	Other Node Configuration Parameters	156
B.5	MySQL Server Option and Variable Reference for MySQL Cluster	157
	Index	167

Preface and Legal Notices

This is the User Manual for the *MySQL™ Cluster Manager*, version 1.3.6. It documents the *MySQL Cluster Manager Agent* and *MySQL Cluster Manager Client* software applications which can be used to administer *MySQL Cluster*, a version of the *MySQL Database System* (referred to hereafter as “MySQL Server” or simply “MySQL”) that incorporates the [NDB](#) storage engine for high availability and data redundancy in a distributed computing environment.

This manual does contain certain basic information about MySQL and MySQL Cluster; however, it is not in any way intended as an exhaustive reference for either of these products.

Current versions of MySQL Cluster compatible with MySQL Cluster Manager are based on versions 7.2, 7.3, and 7.4 of the [NDB](#) storage engine; these versions of MySQL Cluster are known as “MySQL Cluster NDB 7.2,” “MySQL Cluster NDB 7.3,” and “MySQL Cluster NDB 7.4,” respectively. MySQL Cluster functionality varies between MySQL Cluster releases; MySQL Cluster Manager cannot supply or emulate MySQL Cluster features that are not present in the version of the MySQL Cluster software in use.

For complete information about MySQL Server and MySQL Cluster, please refer to the appropriate version of the *MySQL Manual*: For information about MySQL Cluster NDB 7.3 and 7.4, see [MySQL NDB Cluster 7.3 and NDB Cluster 7.4](#).

Licensing information. This product may include third-party software, used under license. See the [MySQL Cluster Manager 1.3 License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this release.

Legal Notices

Copyright © 2009, 2021, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<https://www.oracle.com/corporate/accessibility/>.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab>.

Chapter 1 Overview of MySQL Cluster Manager

Table of Contents

1.1 MySQL Cluster Manager Terminology	1
1.2 MySQL Cluster Manager Architecture	2

This chapter provides an overview of MySQL Cluster Manager, as well as its architecture, purpose, and capabilities.

1.1 MySQL Cluster Manager Terminology

This section provides definitions of key terms used to describe MySQL Cluster Manager and its components in this manual and in other documentation relating to MySQL Cluster Manager and MySQL Cluster.

Site.

A set of hosts on which MySQL Cluster processes to be managed by MySQL Cluster Manager are located. A site can include one or more clusters.

Cluster.

A MySQL Cluster deployment. A cluster consists of a set of MySQL Cluster processes running on one or more hosts. A minimal cluster is usually considered to include one management node, two data nodes, and one SQL node. A typical production cluster may have one or two management nodes, several SQL nodes, and 4 or more data nodes. The exact numbers of data and SQL nodes can vary according to data size, type and rating of hardware used on the hosts, expected throughput, network characteristics, and other factors; the particulars are beyond the scope of this document, and you should consult [MySQL NDB Cluster 7.3](#) and [NDB Cluster 7.4](#), for more specific information and guidelines.

Host.

A computer. The exact meaning depends on the context:

- A computer where one or more MySQL Cluster processes are run. In this context, we sometimes refer more specifically to a *cluster host*.

The number of cluster processes and number of cluster hosts may be, but are not necessarily, the same.

- A computer where an instance of the MySQL Cluster Manager agent runs.

In order to run a MySQL Cluster using MySQL Cluster Manager, the MySQL Cluster Manager agent must be running on each host where cluster processes are to be run. In other words, when using MySQL Cluster Manager, all cluster hosts must also be MySQL Cluster Manager agent hosts (although the reverse is not necessarily true). Therefore, you should understand that anytime we use the term *host*, we are referring to a host computer in both of the senses just given.

Process.

In the context of MySQL Cluster, a process (more specifically, a cluster process) is a MySQL Cluster node, of one of the following 3 types: management node (`ndb_mgmd`), data node (`ndbd` or `ndbmt.d`), or SQL node (`mysqld`). For more information about these node types and their functions in a cluster, see [NDB Cluster Core Concepts](#), and [NDB Cluster Nodes, Node Groups, Fragment Replicas, and Partitions](#).

Package.

A copy of the MySQL cluster software. This should include the binary executables needed to run the cluster processes of the desired types on a given host. The simplest way to make sure that this is done is to place a copy of the entire MySQL Cluster distribution on each computer that you intend to use as a cluster host.

Configuration attribute.

A value whose setting affects cluster operations in a clearly defined and measurable way. When running MySQL Cluster manually, configuration is accomplished using cluster configuration parameters, MySQL server options, and MySQL system and status variables; MySQL Cluster Manager masks the differences between these, providing a unified view of them; see [Configuration attributes](#), for more information.

Agent.

A MySQL Cluster Manager process that runs on each cluster host, responsible for managing the cluster processes running on that host.

Client.

The MySQL Cluster Manager client is a software application that allows a user to connect to MySQL Cluster Manager and perform administrative tasks, such as (but not limited to): creating, starting, and stopping clusters; obtaining cluster and cluster process status reports; getting cluster configuration information and setting cluster configuration attributes.

1.2 MySQL Cluster Manager Architecture

This section provides an architectural overview of MySQL Cluster Manager, its components, and their deployment.

MySQL Cluster Manager is a distributed client/server application consisting of two main components. The MySQL Cluster Manager agent is a set of one or more agent processes that manage MySQL cluster nodes, and the MySQL Cluster Manager client provides a command-line interface to the agent's management functions.

Agent.

The MySQL Cluster Manager agent is comprised of the set of all MySQL Cluster Manager agent processes running on the hosts making up a given management site. A MySQL Cluster Manager agent process is a daemon process which runs on each host to be used in the cluster. In MySQL Cluster Manager, there is no single central server or process; all agents collaborate in managing a cluster as a whole. This means that any connected agent can be used to carry out tasks that effect the entire cluster.

Each agent process is responsible for managing the MySQL Cluster nodes running on the host where the agent is located. MySQL Cluster management and SQL nodes are managed directly by the MySQL Cluster Manager agent; cluster data nodes are managed indirectly, using the cluster management nodes.

Management responsibilities handled by the MySQL Cluster Manager agent include the following:

- Starting, stopping, and restarting cluster nodes
- Cluster configuration changes
- Cluster software upgrades
- Host and node status reporting
- Recovery of failed cluster nodes

Creating, performing initial configuration of, or starting a cluster, requires that agent processes be running on all cluster hosts. Once the cluster has been started, it continues to run even if one or more agent

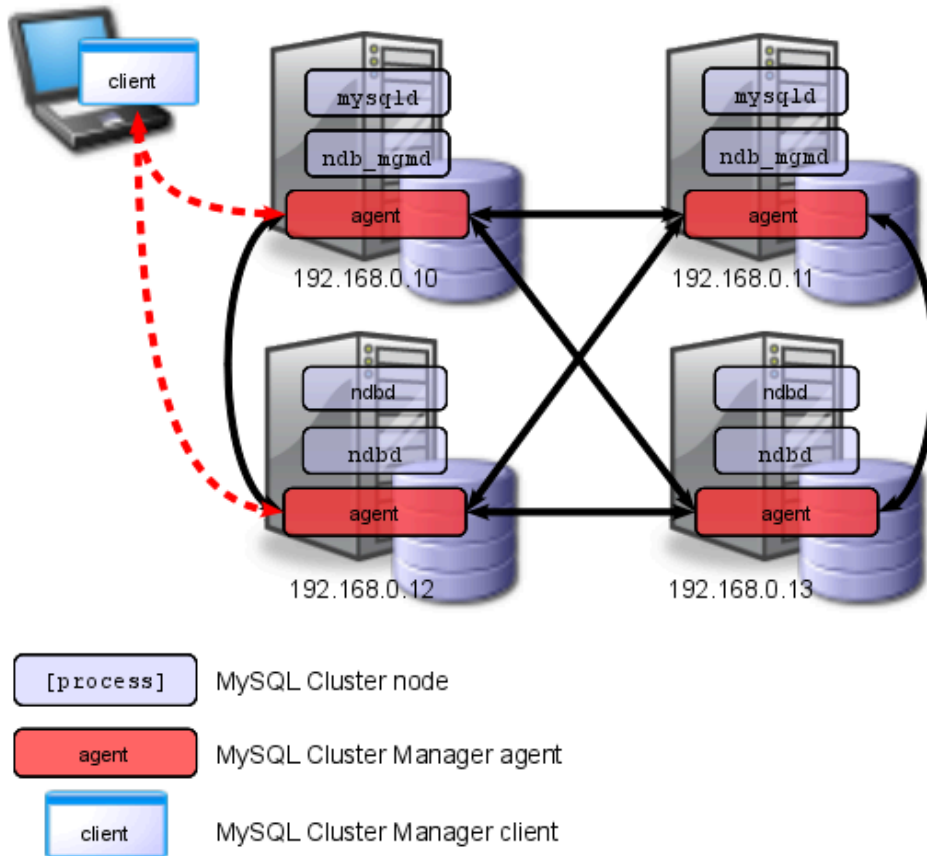
processes fail. However, any failed agent processes must be restarted before you can perform additional cluster management functions.

Client.

A MySQL Cluster Manager client is a software application used to access an MySQL Cluster Manager agent. In MySQL Cluster Manager, the client is actually nothing more than the `mysql` command-line client, started with the options that are necessary for it to connect to an MySQL Cluster Manager agent. MySQL Cluster Manager 1.3.6 and later releases include an `mcm` client for ease of use; this client consists of a script that acts as a wrapper for the `mysql` client with these options.

By way of example, we show how MySQL Cluster Manager would be deployed for use with a MySQL Cluster running on 4 host computers. This is illustrated in the following diagram:

Figure 1.1 MySQL Cluster Manager Deployment



In this example cluster, two of the hosts each house a management server (`ndb_mgmd`) and an SQL node (`mysql`); the other 2 hosts each house two data nodes (`ndbd`). However, regardless of the distribution of cluster nodes among the hosts, a MySQL Cluster Manager agent process must be running on each host.

A MySQL Cluster Manager client can be used to access the agent from any of the hosts making up the management site to which the cluster belongs. In addition, the client can be used on any computer that has a network connection to at least 1 of the hosts where an agent process is running. The computer where the client itself runs is not required to be one of these hosts. The client can connect to and use different agent

processes on different hosts within the management site, at different times, to perform cluster management functions.

Chapter 2 MySQL Cluster Manager Installation, Configuration, Cluster Setup

Table of Contents

2.1 Obtaining MySQL Cluster Manager	5
2.2 Supported Platforms and MySQL Cluster Versions	5
2.3 MySQL Cluster Manager Installation	6
2.3.1 Installing MySQL Cluster Manager on Unix Platforms	6
2.3.2 Installing MySQL Cluster Manager on Windows Platforms	9
2.3.3 Setting the MySQL Cluster Manager Agent User Name and Password	18
2.4 MySQL Cluster Manager Configuration File	19
2.5 Upgrading MySQL Cluster Manager	20

This chapter discusses basic installation and configuration of the MySQL Cluster Manager Management Agent, connecting to the agent with the MySQL Cluster Manager client, and the basics of creating or importing a cluster using MySQL Cluster Manager.

2.1 Obtaining MySQL Cluster Manager

MySQL Cluster Manager is available only through commercial license. To learn more about licensing terms, and to obtain information about where and how to download MySQL Cluster Manager, visit <https://www.mysql.com/products/cluster/mcm/>, or contact your Oracle representative.

2.2 Supported Platforms and MySQL Cluster Versions

For a list of platforms supported by MySQL Cluster Manager 1.3.6, see *Supported Platforms: MySQL Cluster Manager* at <https://www.mysql.com/support/supportedplatforms/cluster-manager.html>, or contact your Oracle representative.

MySQL Cluster Manager 1.3.6 supports the following MySQL Cluster release versions:

- MySQL Cluster NDB 7.1 (see MySQL Cluster NDB 7.1 in the *MySQL 5.1 Reference Manual*), beginning with MySQL Cluster NDB 7.1.4
- MySQL Cluster NDB 7.2 (see MySQL Cluster NDB 7.2 in the *MySQL 5.5 Reference Manual*), beginning with MySQL Cluster NDB 7.2.4
- MySQL Cluster NDB 7.3 (see [MySQL NDB Cluster 7.3 and NDB Cluster 7.4](#)), beginning with MySQL Cluster NDB 7.3.2
- MySQL Cluster NDB 7.4 (see [MySQL NDB Cluster 7.3 and NDB Cluster 7.4](#)), beginning with MySQL Cluster NDB 7.4.4

Prior to installation, you must obtain the correct build of MySQL Cluster Manager for your operating system and hardware platform. For Unix platforms, MySQL Cluster Manager is delivered as a Unix `.tar.gz` archive, whose name is in the format of `mcm-1.3.6-cluster-7.4.6-linux-distro-arch.tar.gz`. For Windows platforms, an MSI installer file is provided. All MySQL Cluster Manager 1.3.6 packages include MySQL Cluster NDB 7.4.6.

When selecting the appropriate installer for your operating system and hardware, keep in mind that 32-bit programs can normally be run on 64-bit platforms, but that the reverse is not true.

2.3 MySQL Cluster Manager Installation

Installation of the MySQL Cluster Manager agent and client programs varies according to platform. On Unix platforms, you must extract the binaries and other files from a `.tar.gz` archive and copy them to the correct locations manually. For Windows platforms, an MSI installer is provided which largely automates this process. Installation of MySQL Cluster Manager using each of these methods is covered in the next two sections.



Note

License keys were required in order to use some very early releases of MySQL Cluster Manager. They are not needed in order to use recent versions, including MySQL Cluster Manager 1.3.6.

2.3.1 Installing MySQL Cluster Manager on Unix Platforms

Install the MySQL Cluster Manager agent on Linux and similar platforms by following the instructions below.

Extract the MySQL Cluster Manager 1.3.6 program and other files from the distribution archive.

You must install a copy of MySQL Cluster Manager on each computer that you intend to use as a MySQL Cluster host. In other words, you need to install MySQL Cluster Manager on each host that is a member of a MySQL Cluster Manager management site. For each host, you should use the MySQL Cluster Manager build that matches that computer's operating system and processor architecture.

On Linux systems, you can unpack the archive using the following command, using `mcm-1.3.6-cluster-7.4.6-linux-sles11-x86-64bit.tar.gz` as an example (the actual filename will vary according to the MySQL Cluster Manager build that you intend to deploy):

```
shell> tar -zxvf mcm-1.3.6-cluster-7.4.6-linux-sles11-x86-64bit.tar.gz
```

This command unpacks the archive into a directory having the same name as the archive, less the `.tar.gz` extension. The top-level directory within the archive is named `mcm-1.3.6`.



Important

Because the Solaris version of `tar` cannot handle long filenames correctly, the MySQL Cluster Manager program files may be corrupted if you try to use it to unpack the MySQL Cluster Manager archive. To get around this issue on Solaris operating systems, you should use GNU `tar` (`gtar`) rather than the default `tar` supplied with Solaris. On Solaris 10, `gtar` is often already installed in the `/usr/sfw/bin` directory, although the `gtar` executable may not be included in your path. If `gtar` is not present on your system, please consult the [Solaris 10 system documentation](#) for information on how to obtain and install it.

In general, the location where you place the unpacked MySQL Cluster Manager directory and the name of this directory can be arbitrary. However, we recommend that you use a standard location for optional software, such as `/opt` on Linux systems, and that you name the directory using the 1.3.6 version number. (This facilitates subsequent upgrades.) On a typical Linux system you can accomplish this task like this:

```
shell> cd mcm-1.3.6-cluster-7.4.6-linux-sles11-x86-64bit
shell> mv mcm-1.3.6 /opt/mcm-1.3.6
```

For ease of use, we recommend that you put the MySQL Cluster Manager files in the same directory on each host where you intend to run it.

Contents of the MySQL Cluster Manager Unix Distribution Archive.

If you change to the directory where you placed the extracted MySQL Cluster Manager archive and list the contents, you should see something similar to what is shown here:

```
shell> cd /opt/mcm-1.3.6
shell> ls
bin cluster etc lib libexec licenses share var
```

These directories are described in the following table:

Table 2.1 Contents of the MySQL Cluster Manager Unix distribution archive, by directory

Directory	Contents
<code>bin</code>	MySQL Cluster Manager agent startup scripts
<code>cluster</code>	Contains the MySQL Cluster NDB 7.4.6 binary distribution
<code>etc</code>	Contains the agent configuration file (<code>mcmd.ini</code>)
<code>etc/init.d</code>	Init scripts
<code>lib</code> and subdirectories	Libraries needed to run the MySQL Cluster Manager agent
<code>libexec</code>	MySQL Cluster Manager agent and client executables
<code>licenses/lgpl</code>	An archive containing source code (including licensing and documentation), for <code>glib 2.1</code>
<code>share/doc/mcmd</code>	<code>README.txt</code> file
<code>var</code>	XML files containing information needed by MySQL Cluster Manager about processes, attributes, and command syntax

Normally, the only directories of those shown in the preceding table that you need be concerned with are the `bin` and `etc` directories.

For MySQL Cluster Manager 1.3.6 distributions that include MySQL Cluster, the complete MySQL Cluster NDB 7.4.6 binary distribution is included in the `cluster` directory. Within this directory, the layout of the MySQL Cluster distribution is the same as that of the standalone MySQL Cluster binary distribution. For example, MySQL Cluster binary programs such as `ndb_mgmd`, `ndbd`, `ndbmt`, and `ndb_mgm` can be found in `cluster/bin`. For more information, see [MySQL Installation Layout for Generic Unix/Linux Binary Package](#), and [Installing an NDB Cluster Binary Release on Linux](#), in the *MySQL Manual*.

If you wish to use the included MySQL Cluster software, it is recommended that you move the `cluster` directory and all its contents to a location outside the MySQL Cluster Manager installation directory, such as `/opt/ndb-version`. For example, on a Linux system, you can move the MySQL Cluster NDB 7.4.6 software that is bundled with MySQL Cluster Manager 1.3.6 to a suitable location by first navigating to the MySQL Cluster Manager installation directory and then using a shell command similar to what is shown here:

```
shell> mv cluster /opt/ndb-7.4.6
```



Note

The `mcmd --bootstrap` option uses the included MySQL Cluster binaries in the installation directory's `cluster` directory and does not work if they cannot be found there. To work around this issue, create a symbolic link to the correct directory in its place, like this:

```
shell> ln -s /opt/ndb-7.4.6 cluster
```

After doing this, you can use the `mcm` client commands `add package` and `upgrade cluster` to upgrade any desired cluster or clusters to the new MySQL Cluster software version.



Important

On Linux platforms, do not attempt to install the MySQL Cluster software by the RPM, Debian, or other installation packages for any package management systems. They install MySQL Cluster differently than the binary distribution that comes with the MySQL Cluster Manager archive, and that will cause issue in the future when you try to upgrade your cluster with MySQL Cluster Manager.

The MySQL Cluster Manager agent by default writes its log file as `mcmd.log` in the installation directory. When the agent runs for the first time, it creates a directory where the agent stores its own configuration data; by default, that is `./mcm_data` (relative to the installation directory of MySQL Cluster Manager). The configuration data, log files, and data node file systems for a given MySQL Cluster under MySQL Cluster Manager control, and named `cluster_name`, can be found in `clusters/cluster_name` under this data directory (sometimes also known as the MySQL Cluster Manager *data repository*).

The location of the MySQL Cluster Manager agent configuration file, log file, and data directory can be controlled with `mcmd` startup options or by making changes in the agent configuration file. To simplify upgrades of MySQL Cluster Manager, we recommend that you change the data repository to a directory outside the MySQL Cluster Manager installation directory, such as `/var/opt/mcm`. See [Section 2.4, “MySQL Cluster Manager Configuration File”](#), and [Section 3.2, “Starting and Stopping the MySQL Cluster Manager Agent”](#), for more information.

On Linux and other Unix-like systems, you can set up the MySQL Cluster Manager agent to run as a daemon, using the init script that is supplied with the MySQL Cluster Manager distribution. To do this, follow the steps listed here:

1. Copy the file `/etc/init.d/mcmd` under the MySQL Cluster Manager installation directory to your system's `/etc/init.d/` directory (or equivalent). On a typical Linux system, you can do this using the following command in the system shell, where `mcmdir` is the MySQL Cluster Manager installation directory:

```
shell> cd mcmdir/etc/init.d
shell> cp mcmd /etc/init.d/mcmd
```

2. Make sure that this file has appropriate permissions and is executable by the user account that runs MySQL Cluster Manager. On a typical Linux system, this can be done by executing commands in your system shell similar to those shown here:

```
shell> chown mcmuser /etc/init.d/mcmd
shell> chmod 755 /etc/init.d/mcmd
```

Be sure to refer to your operating system documentation for exact information concerning the commands needed to perform these operations, as they may vary between platforms.

3. Open the file `/etc/init.d/mcmd` in a text editor. Here, we show a portion of this file, in which we have highlighted the two lines that need to be updated:

```
MCMD_SERVICE="mcmd"
MCMD_PSERVICE="MySQL Cluster Manager"
MCMD_ROOTDIR="@@MCMD_ROOTDIR@@
MCMD_BIN="$MCMD_ROOTDIR/bin/mcmd"
MCMD_CONFIG="$MCMD_ROOTDIR/etc/mcmd.ini"
```

```
# Run service as non-root user
MCMD_USER=@@MCMD_USER@@
SU="su --login $MCMD_USER --command"
```

In the first of these lines, replace the placeholder `@@MCMD_ROOTDIR@@` with the complete path to the MySQL Cluster Manager installation directory. In the second of these lines, replace the placeholder `@@MCMD_USER@@` with the name of the system user that runs the MySQL Cluster Manager agent (note that this must *not* be the system `root` account). Save the edited file.

The MySQL Cluster Manager agent should now be started automatically whenever the system is restarted.

When the agent is configured as a daemon, cluster processes are started automatically when the agent is restarted, as long as the cluster was running when the agent shut down. *Note that `StopOnError` must be disabled (set to 0) for all data nodes in order for this to work.* If the cluster was stopped when the agent shut down, it is necessary to supply a script which waits for the agent to complete its startup and recovery phases, and then, when the agent is ready, starts the cluster using a command such as `mcmdir/bin/mcm -e 'start cluster --background cluster_name;'`.

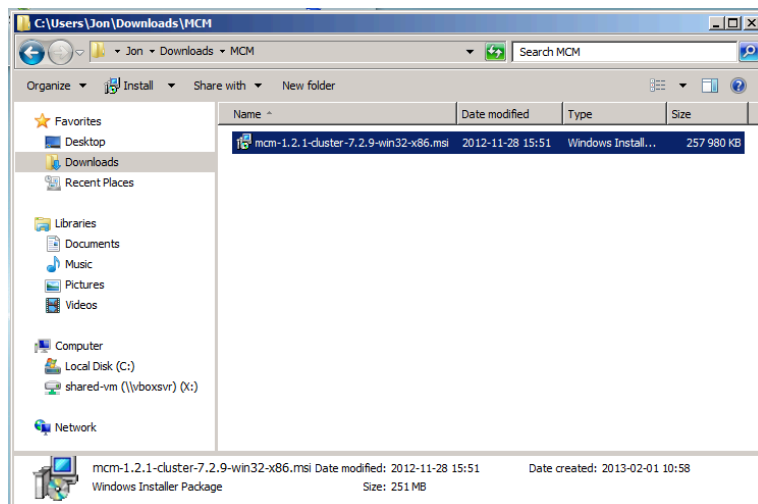
2.3.2 Installing MySQL Cluster Manager on Windows Platforms

To install MySQL Cluster Manager 1.3.6 on Windows platforms, you should first have downloaded the MSI installer file `mcm-1.3.6-cluster-7.4.6-win32-x86.msi` (see [Section 2.1, “Obtaining MySQL Cluster Manager”](#)). 1.3.6 for Windows is 32-bit, as is the bundled MySQL Cluster NDB 7.4.6; these run on both 32-bit and 64-bit versions of Windows.

As mentioned elsewhere (see, for example, [Section 3.4.1, “Creating a MySQL Cluster with MySQL Cluster Manager”](#)), you must install a copy of MySQL Cluster Manager on each computer where you intend to host a MySQL Cluster node. Therefore, the following procedure must be performed separately on each host computer. For ease of installations and upgrades on multiple machines, it is recommended that you install MySQL Cluster Manager to the same location on each host. This is `C:\Program Files\MySQL\MySQL Cluster Manager 1.3.6\` or `C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.3.6\` by default, but it is possible to install MySQL Cluster Manager to an alternate location such as `C:\mcm\`.

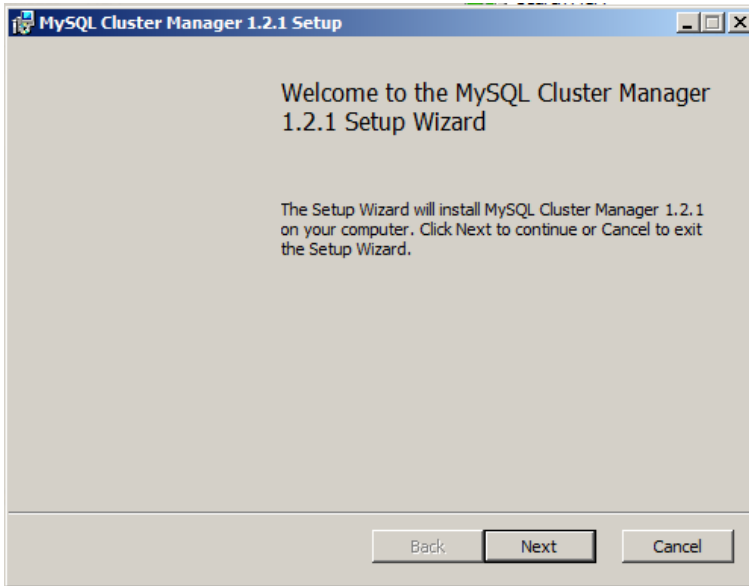
In the discussion that follows, we assume that you have downloaded the MySQL Cluster Manager 1.2.1 MSI installer as user `jon` to this user's Downloads directory (`C:\Users\jon\Downloads\MCM`) on a 64-bit Windows system. Navigate to this directory in Windows Explorer, as shown here:

Figure 2.1 MySQL Cluster Manager MSI Installer: Download



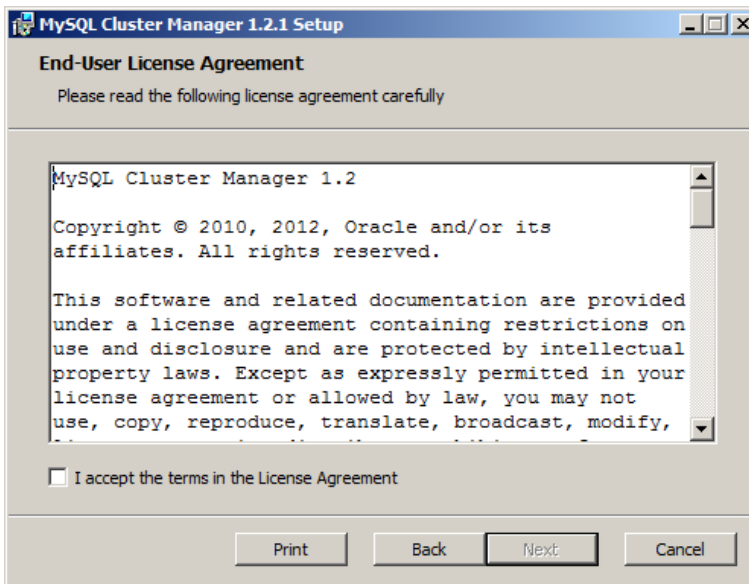
To run the installer, double-click on the file icon in Windows Explorer. Some versions of Windows also provide an **Install** item in the Windows Explorer menu that can be used to run the installer. When you start the installer, you may see a Windows **Security Warning** screen. If you obtained the installer from a trusted source and know that it has not been tampered with, choose **Run** from the dialog, which allows the installer to continue to the **Welcome** screen, as shown here:

Figure 2.2 MySQL Cluster Manager MSI Installer: Welcome Screen

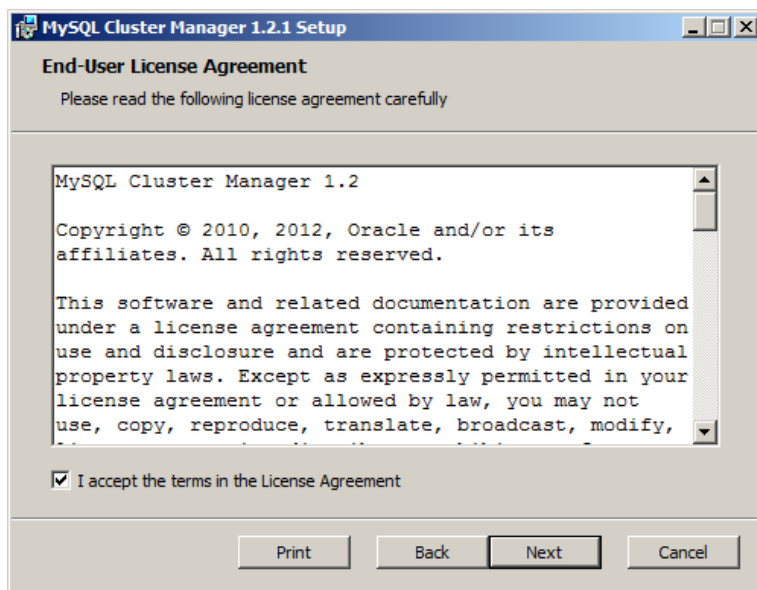


Click the **Next** button to continue to the **License Agreement** screen, as shown in the next figure:

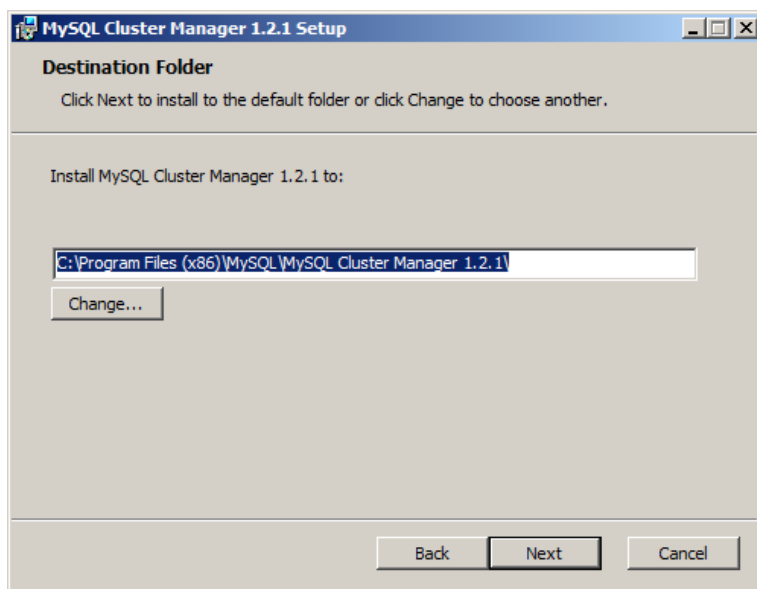
Figure 2.3 MySQL Cluster Manager MSI Installer: License Agreement



You should read the license text in the text area, and when you have done so, check the box labelled **I accept the terms in the License Agreement**. Until you have checked the box, you cannot complete the MySQL Cluster Manager installation; it is possible only to print the license, go back to the previous screen, or cancel the installation (using the buttons labelled **Print**, **Back**, and **Cancel**, respectively). Checking the box enables the **Next** button, as shown here:

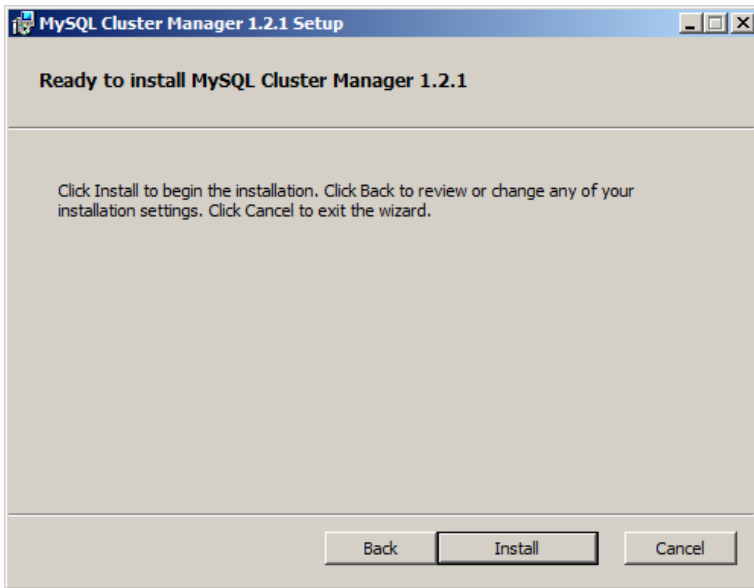
Figure 2.4 MySQL Cluster Manager MSI Installer: Accepting License Agreement

Click the **Next** button to continue to the **Destination Folder** screen, where you can choose the installation directory. The next figure shows the **Destination Folder** screen with the default location. On English-language 32-bit Windows systems, this is `C:\Program Files\MySQL\MySQL Cluster Manager version\`; on English 64-bit Windows systems, the default is `C:\Program Files (x86)\MySQL\MySQL Cluster Manager version\`. (In this example, we are installing MySQL Cluster Manager 1.2.1 on a 64-bit system, so the default is `C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.2.1\`.) You can click the **Change** button to change the directory where MySQL Cluster Manager should be installed; the default directory is adequate for most cases.

Figure 2.5 MySQL Cluster Manager MSI Installer: Destination Folder

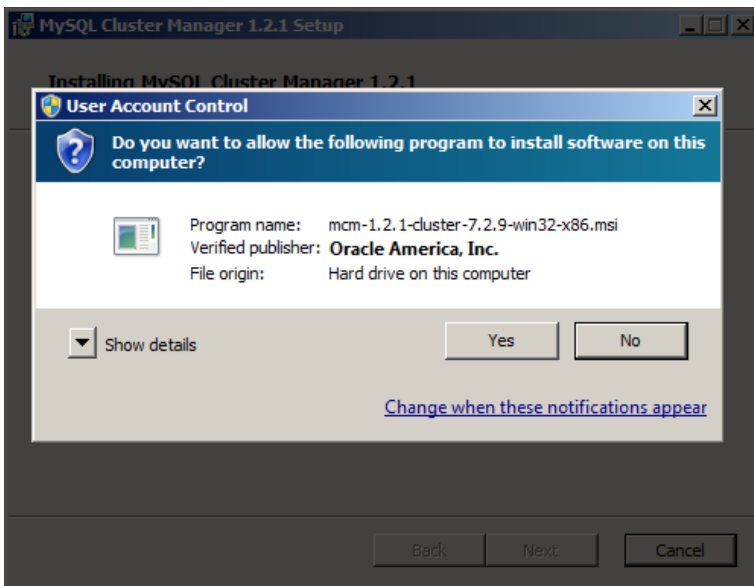
Once you have selected the destination directory, the installer has gathered all the information that it requires to perform the installation. Click **Next** to continue to the **Ready** screen, shown here:

Figure 2.6 MySQL Cluster Manager MSI Installer: Ready to Install

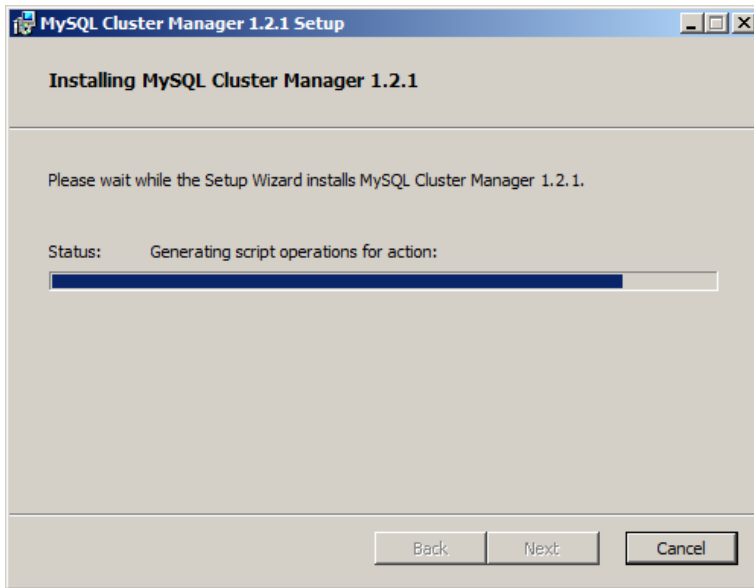


Click the **Install** button to install MySQL Cluster Manager. As the installer begins to copy files and perform other tasks affecting the system, you may see a warning dialog from Windows User Access Control, as shown here:

Figure 2.7 MySQL Cluster Manager MSI Installer: User Account Control



If this occurs, click the **Yes** button to allow the installation to continue. A **Setup Wizard** screen with a progress bar is displayed while the installer runs, as shown in the next figure:

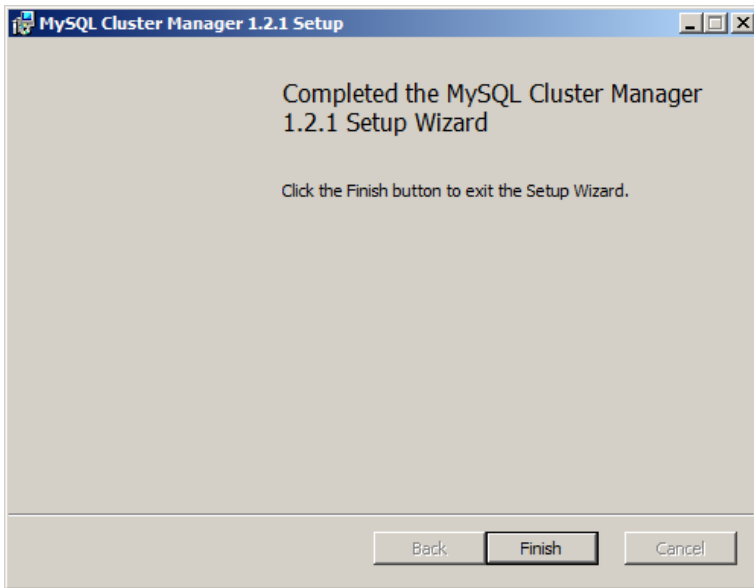
Figure 2.8 MySQL Cluster Manager MSI Installer: Setup Wizard

The **Setup Wizard** may require several minutes to copy all of the necessary files for MySQL Cluster Manager 1.3.6 and MySQL Cluster NDB 7.4.6 to the installation directory and to perform other required changes.

**Note**

The MySQL Cluster Manager 1.3.6 installer places MySQL Cluster NDB 7.4.6 in the `cluster` directory under the installation directory. (By default, this is `C:\Program Files\MySQL\MySQL Cluster Manager 1.3.6\cluster` or `C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.3.6\cluster`.) The location of the MySQL Cluster binaries is not separately configurable using the MySQL Cluster Manager installer.

When the **Setup Wizard** finishes, the installer displays the **Installation Completed** screen, as shown here:

Figure 2.9 MySQL Cluster Manager MSI Installer: Installation Completed

MySQL Cluster Manager 1.3.6 has now been installed to the destination directory; click the **Finish** button to exit the installer.

2.3.2.1 Installing the MySQL Cluster Manager Agent as a Windows Service

After installing the MySQL Cluster Manager Agent as a Windows service, you can start and stop the agent using the Windows Service Manager. The installation also configures the agent to start automatically whenever Windows starts, and to shut down safely whenever Windows shuts down.

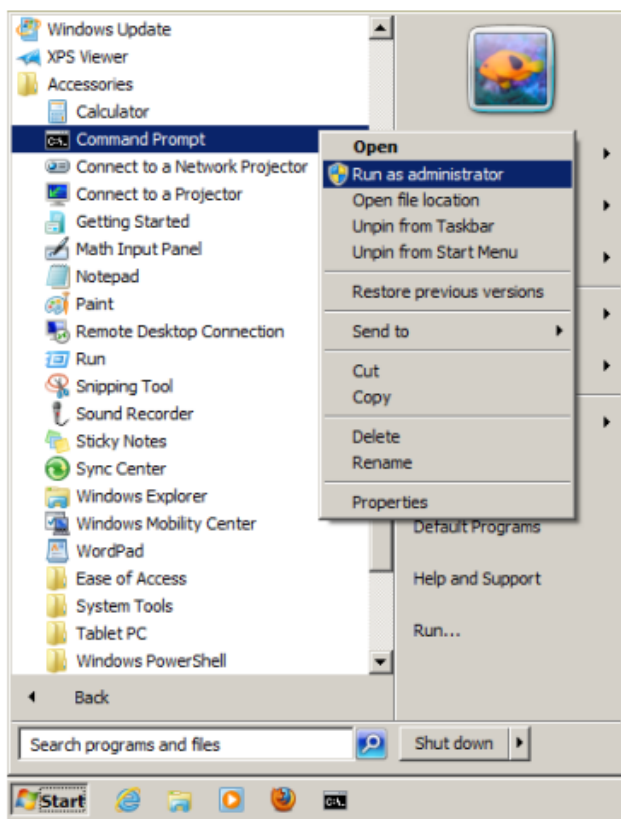


Note

The Windows service can be used to control the running of MySQL Cluster Manager agents on a single host only. To shut down agents on multiple hosts, you can use the `stop agents` command in the MySQL Cluster Manager client.

The installation is performed using the command prompt (`cmd.exe`); as with installing or removing any Windows service, it must also be done as a user having sufficient permissions, such as the system Administrator account.

If the account you are currently using has Administrator privileges, you can simply start `cmd.exe`. Otherwise, you must run the command prompt program as the Administrator. To do this, first locate a shortcut to the command prompt. On most Windows systems, you can do this using the Start Menu. Find **Programs** (or **All Programs**, in some Windows versions), then navigate to **Accessories**. Under **Accessories**, right-click on the **Command Prompt** menu item. From the context menu that appears, select **Run as Administrator**. You can see how this looks on a typical Windows system in the next figure.

Figure 2.10 Installing the MySQL Cluster Manager Agent as a Windows Service: Command Prompt

If a Windows UAC dialog referring to `cmd.exe` appears, click **Yes** to allow the command prompt to run as Administrator and thus to continue. You should now have a command prompt window open on your desktop, running a session with Administrator privileges.

To install the MySQL Cluster Manager agent as a service, we use the `SC CREATE` command. This command allows us to specify a name for the service (for use with `SC START` and `SC STOP` or `NET START` and `NET STOP` commands), a display name (to be shown in the Service Manager), a startup mode (automatic or manual start), and a path to the executable to be run as a service. (Use `mcmd-svc.exe` rather than `mcmd.exe` as the executable.) The path must also include any arguments needed by the program; in the case of MySQL Cluster Manager, `mcmd-svc.exe` must be told where to find its configuration file using the `--defaults-file` option. Both of these paths must be absolute.



Important

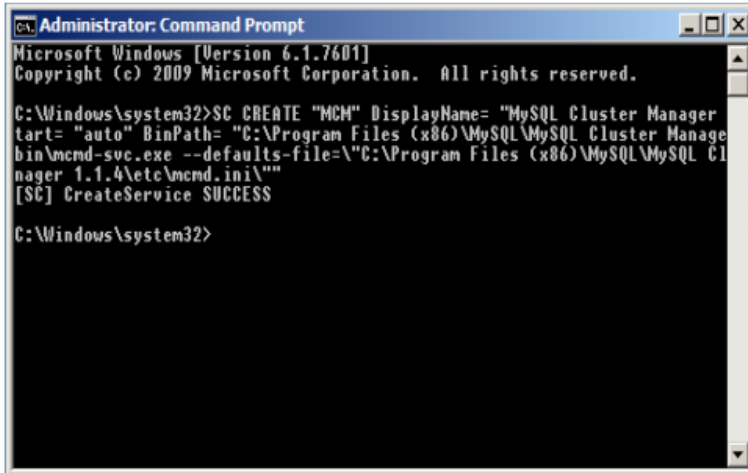
Installation of the MySQL Cluster Manager agent as a service is recommended. However, you should *not* install MySQL Cluster processes (`ndb_mgmd.exe`, `ndbd.exe`, `ndbmttd.exe`, `mysqld.exe`) as services on Windows hosts to be used as MySQL Cluster nodes under management by MySQL Cluster Manager, since the MySQL Cluster Manager agent itself controls MySQL Cluster nodes independently of the Windows Service Manager.

Assume that you have installed MySQL Cluster Manager to the default location for 64-bit Windows systems `C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.3.6\` (`C:\Program Files\MySQL\MySQL Cluster Manager 1.3.6\` on 32-bit Windows systems), and that its configuration file is located in `C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.3.6\etc`. Then the following command installs MySQL Cluster Manager as a service named `mcm`, with the display name “MySQL Cluster Manager 1.3.6”:

```
C:\> SC CREATE
"MCM" DisplayName= "MySQL Cluster Manager 1.3.6" Start= "auto"
BinPath= "C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.3.6\bin\mcmd-svc.exe
--defaults-file="C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.3.6\etc\mcmd.ini\"""
[SC] CreateService SUCCESS
C:\>
```

This command can be quite long. For enhanced legibility, we have broken it across several lines, but you should always enter it on a single line, allowing it to wrap naturally, similar to what is shown here:

Figure 2.11 Installing the MySQL Cluster Manager Agent as a Windows Service: Command



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>SC CREATE "MCM" DisplayName= "MySQL Cluster Manager
tart= "auto" BinPath= "C:\Program Files (x86)\MySQL\MySQL Cluster Manage
bin\mcmd-svc.exe --defaults-file="C:\Program Files (x86)\MySQL\MySQL Cl
nager 1.1.4\etc\mcmd.ini\"""
[SC] CreateService SUCCESS

C:\Windows\system32>
```

In addition, you should keep in mind that the spaces after the equal signs following the `DisplayName`, `Start`, and `BinPath` arguments are required.

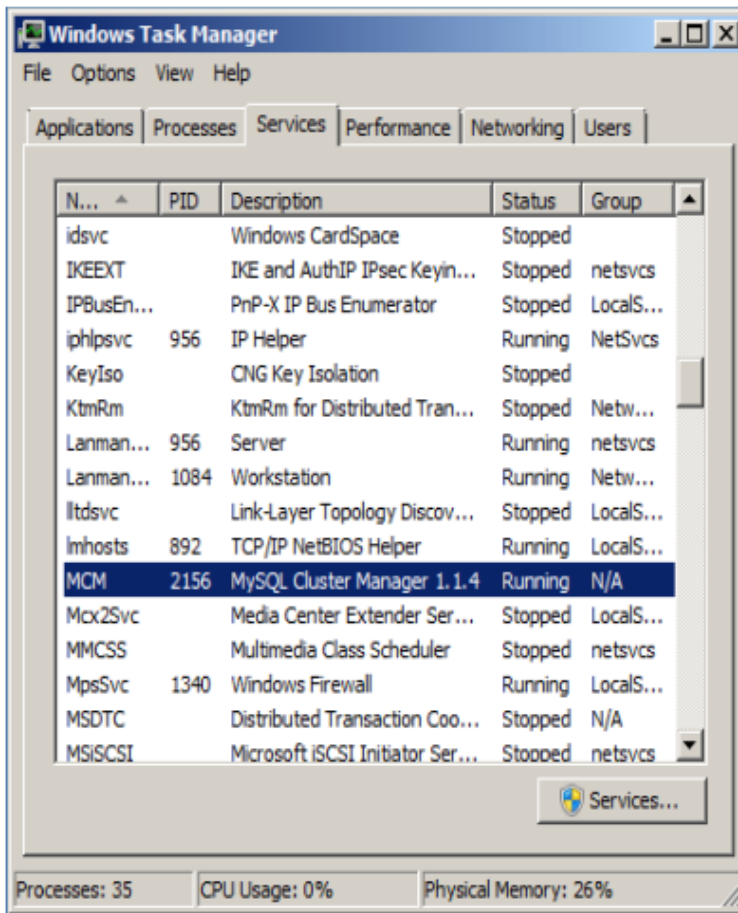
Starting and stopping the MySQL Cluster Manager agent Windows service. After installing the service successfully, you can start and stop the service manually, if the need arises, with the `SC START` and `SC STOP` commands.

```
C:\> SC START MCM
C:\> SC STOP MCM
```

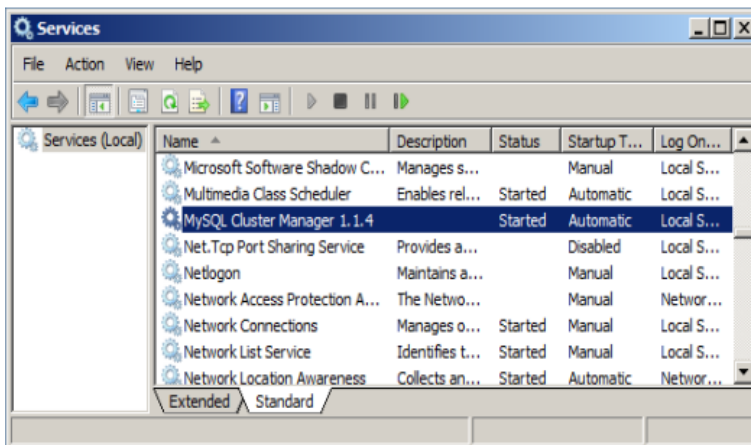
Alternatively, use the `NET START` and `NET STOP` commands:

```
C:\Windows\system32>NET START MCM
C:\Windows\system32>NET STOP MCM
```

Once the service is installed, the MySQL Cluster Manager agent starts automatically whenever Windows is started. You can verify that the service is running with the Windows Task Manager. Open the Task Manager, and switch to the **Services** tab if it is not already displayed. If the MySQL Cluster Manager agent is running, you can find it in the list of services under `MCM` in the `Name`, column and `MySQL Cluster Manager 1.3.6` in the `Description` column, as shown here:

Figure 2.12 Installing the MySQL Cluster Manager Agent as a Windows Service: Services Tab

You can also verify if the service is running using the Windows Service Manager, as shown here:

Figure 2.13 Installing the MySQL Cluster Manager Agent as a Windows Service: Service Manager

The Service Manager also allows you to start, stop, or pause the MySQL Cluster Manager agent service manually using a GUI.

**Note**

When first installing the MySQL Cluster Manager agent as a service, the service is not started automatically until Windows is started. If you do not wish to restart Windows, then you must start the service manually using either `SC START` or `NET START` on the command line or the graphical control provided in the Windows Service Manager.

You can remove the service using the `SC DELETE` command and the name of the service—in this case `MCM`—that was used in the `SC CREATE` command, as shown here:

Figure 2.14 Removing the MySQL Cluster Manager Agent from Windows Services

```

Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>NET STOP MCM
The MySQL Cluster Manager 1.1.4 service is stopping.
The MySQL Cluster Manager 1.1.4 service was stopped successfully.

C:\Windows\system32>SC DELETE MCM
[SC] DeleteService SUCCESS

C:\Windows\system32>
  
```

If the service is running at the time that `SC DELETE` is executed, the removal of the service takes effect the next time the service is stopped. In such a case, you must stop the previous instance of the service manually, and allow it to be removed, before you can reinstall the service.

Once you have installed the MySQL Cluster Manager agent and the service is running correctly, you are ready to connect to it using the MySQL Cluster Manager client. See [Section 3.3, “Starting the MySQL Cluster Manager Client”](#), for information about how to do this.

2.3.3 Setting the MySQL Cluster Manager Agent User Name and Password

Normally it is not necessary to alter the user name or password used by the user agent to administer `mysqld` processes. However, if you should wish to do so, you can change either or both of these, using the procedure outlined here:

1. Stop all agents. (You can use `stop agents` for this purpose.)
2. Update the `etc/mcmd.ini` agent configuration file. Set a new password by uncommenting the line containing `manager-password=` and adding the new password as its value; set a new administrative user account name by uncommenting the line containing `manager-username=` and setting its value to the new user name. See [Section 3.1, “mcmd, the MySQL Cluster Manager Agent”](#), for more information about these options.
3. For each `mysqld` do the following:
 - a. Log in (using the `mysql` client) as the MySQL `root` user

- b. If you are changing the user name, do this first, using the following statement, where *olduser* is the current user name and *newuser* is the new *manager-user* that you set previously in *etc/mcmd.ini*:

```
RENAME USER 'olduser'@'127.0.0.1' TO 'newuser'@'127.0.0.1';
```

If you are changing the user name for the first time, use *mcmd* for *olduser*. In addition, you should use *127.0.0.1* for the host name (and not *localhost*).

- c. Execute the following statement, where *newuser* is the new user name, and *newpass* is the new password:

```
SET PASSWORD FOR 'newuser'@'127.0.0.1' = PASSWORD('newpass');
```

Use *mcmd* for the user name if you have not changed it—that is, if *manager-user* has been left unset in *mcmd.ini*. Use *127.0.0.1* for the host name (and not *localhost*).

- d. Issue a `FLUSH PRIVILEGES` statement.
4. Restart the agents. All agents should now be using the new password for the *mcmd* accounts on the MySQL servers (*mysqld* processes).

2.4 MySQL Cluster Manager Configuration File

Before starting the MySQL Cluster Manager agent, you should make any necessary changes to the `[mcmd]` section of the agent configuration file, supplied with MySQL Cluster Manager as *etc/mcmd.ini* in the installation directory. On Linux and similar operating systems, you can edit this file in place; on Windows, it is recommended that you save a copy of this file to a convenient location for which the path does not contain any spaces, such as `C:\mcm\data`.

A `#` character in the MySQL Cluster Manager configuration file indicates the beginning of a comment; the comment continues to the end of the line.

The version of this file that is supplied with MySQL Cluster Manager reads similarly to what is shown here:

```
[mcmd]
plugins=manager
##manager-port=1862
##agent-uuid=
log-file=mcmd.log
log-level=message
##pid-file=
## Username and password for manager plugin
##manager-username=
##manager-password=
## Top-level directory for manager plugins information stored on disk
##manager-directory=
```

A minimal agent configuration file as used in production might look like this:

```
[mcmd]
plugins=manager
manager-port=1862
```

```
manager-directory=/var/opt/mcm-1.3.6
log-file=/var/opt/mcm/mcmd.log
log-level=warning
```

For more information about these and other options which can be set in the agent configuration file, see [Section 3.1, “mcmd, the MySQL Cluster Manager Agent”](#).

2.5 Upgrading MySQL Cluster Manager

This section discusses upgrading MySQL Cluster Manager from a previous release to the latest 1.3 release (currently 1.3.6), as well as providing basic guidance on upgrading the bundled MySQL Cluster software.

The basic steps for upgrading a MySQL Cluster Manager installation are listed here:

1. Install the new version of the MySQL Cluster Manager software in the desired location.
2. Create a configuration for the new installation such that it uses the previous installation's data.
3. Stop all running MySQL Cluster Manager agent processes on all hosts.
4. Start the new agent processes, ensuring that they use the new configuration created previously.

A more detailed explanation is provided of each of these steps in the next few paragraphs. For purposes of example, we assume an upgrade from an existing installation of MySQL Cluster Manager 1.2.4 to a new installation of MySQL Cluster Manager 1.3.6. For a Linux or other Unix-like system, we assume that these are installed in `/opt/mcm-1.2.4` and `/opt/mcm-1.3.6`, respectively; on Windows, we assume the default directories `C:\Program Files\MySQL\MySQL Cluster Manager 1.2.4\` and `C:\Program Files\MySQL\MySQL Cluster Manager 1.3.6\`.

Step 1: Install new MySQL Cluster Manager version. You can obtain and install a new version of MySQL Cluster Manager in the same way as for a new installation (see [Section 2.1, “Obtaining MySQL Cluster Manager”](#), and [Section 2.3, “MySQL Cluster Manager Installation”](#)), with the additional requirement that you should not attempt to install the new version in the same location as the version which you are currently using.

Step 2: Configure new installation. In order for the new MySQL Cluster Manager agent binaries to manage the same MySQL Cluster instances, they must be able to find the data contained in the agent repository used by the old installation's binaries, which is `../mcm_data` (relative to the installation directory of MySQL Cluster Manager) by default, but can be set using the `manager-directory` directive in `mcmd.ini`.

It is simplest for MySQL Cluster Manager software upgrades if the agent repository and the agent configuration file are located externally to the agent installation directory. Suppose the old version of the agent is installed to `/opt/mcm-1.2.4`, and that it uses the directory `/var/opt/mcm` for its agent repository and `/etc/mcm/mcmd.ini` for its configuration file. In this case, to make the new binaries use the same configuration and repository, it is necessary only to start the new `mcmd` processes such that they use the same configuration file (see [Step 4](#)). Otherwise, you should copy the default configuration file to a location outside the installation directory, such as `/etc/mcm/mcmd.ini`, as shown here:

```
shell> cp -r /opt/mcm-1.2.4/etc/mcmd.ini /etc/mcm/mcmd.ini
```

Then, add the following line to the new copy of the `mcmd.ini` file:

```
manager-directory=/var/opt/mcm
```

After this, you can save and close the file. See also [Section 2.4, “MySQL Cluster Manager Configuration File”](#).

Step 3: Stop all agents. Stop the agent processes using the old binaries on all hosts making up the management installation. You can stop all agents for a given site, for example `mysite`, using the `stop agents` command in the MySQL Cluster Manager client, as shown here:

```
mcm> stop agents mysite;
```

You should execute a `stop agents` command, similar to the one just shown, for each site listed in the output of `list sites`.

Step 4: Start new MySQL Cluster Manager binaries. Start the new `mcmd` agent binaries with the `--defaults-file` option so that it uses the correct configuration file, like this:

```
shell> mcmd --defaults-file=/etc/mcm/mcmd.ini &
```



Note

A majority of the agents (i.e., at least half of the total number plus one) should be started within a period of 10 seconds; otherwise, the lack of a quorum of nodes for decision making might cause the communication among the nodes to break down.

You should now be able to start the `mcm` client from the new installation and perform management tasks as usual. Once the client successfully starts and connects to the agent, you can verify that it is running the correct version of the MySQL Cluster Manager software using the `version` command, as shown here:

```
mcm> version;
+-----+
| Version |
+-----+
| MySQL Cluster Manager 1.3.6 |
+-----+
1 row in set (0.00 sec)
```

Next, check that all hosts, clusters, and processes on all sites are visible to the new `mcm` client, and are operational; for example:

```
mcm> list hosts mysite;
+-----+-----+-----+
| Host | Status | Version |
+-----+-----+-----+
| tonfisk | Available | 1.3.6 |
| flundra | Available | 1.3.6 |
| alpha | Available | 1.3.6 |
| beta | Available | 1.3.6 |
| gamma | Available | 1.3.6 |
+-----+-----+-----+

5 rows in set (0.16 sec)

mcm> list clusters mysite;
+-----+-----+
| Cluster | Package |
+-----+-----+
| mycluster | mypackage |
| yourcluster | mypackage |
+-----+-----+

2 rows in set (2.07 sec)

mcm> show status --cluster mycluster;
+-----+-----+-----+
| Cluster | Status | Comment |
+-----+-----+-----+
```

```

+-----+-----+-----+
| mycluster | fully operational |      |
+-----+-----+-----+
1 row in set (0.01 sec)

mcm> show status --cluster yourcluster;
+-----+-----+-----+
| Cluster   | Status           | Comment |
+-----+-----+-----+
| yourcluster | fully operational |      |
+-----+-----+-----+
1 row in set (0.01 sec)

mcm> show status -r mycluster;
+-----+-----+-----+-----+-----+-----+
| NodeId | Process  | Host      | Status | Nodegroup | Package |
+-----+-----+-----+-----+-----+-----+
| 49     | ndb_mgmd | tonfisk   | running |           | mypackage |
| 1      | ndbd     | tonfisk   | running | 0         | mypackage |
| 2      | ndbd     | flundra   | running | 0         | mypackage |
| 50     | mysqld   | tonfisk   | running |           | mypackage |
| 51     | mysqld   | flundra   | running |           | mypackage |
| 52     | ndbapi   | *tonfisk  | added   |           |           |
| 53     | ndbapi   | *flundra  | added   |           |           |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.08 sec)

mcm> show status -r yourcluster;
+-----+-----+-----+-----+-----+-----+
| NodeId | Process  | Host      | Status | Nodegroup | Package |
+-----+-----+-----+-----+-----+-----+
| 50     | ndb_mgmd | alpha     | running |           | mypackage |
| 5      | ndbd     | beta      | running | n/a       | mypackage |
| 6      | ndbd     | gamma     | running | n/a       | mypackage |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
    
```

See [Chapter 3, Using MySQL Cluster Manager](#), for more information about performing common cluster management tasks with the `mcm` client.

Upgrading MySQL Cluster. Although the MySQL Cluster software typically comes bundled with the MySQL Cluster Manager distribution, it is important to keep in mind that upgrading the MySQL Cluster Manager software does *not* upgrade any existing MySQL Cluster installations. Since the new MySQL Cluster Manager installation uses the same configuration as the old one, the clusters under its control remain unchanged. If you wish to upgrade a cluster using the bundled MySQL Cluster software, you should move the `cluster` directory (see [Contents of the MySQL Cluster Manager Unix Distribution Archive](#)) and all of its contents to a location outside the MySQL Cluster Manager installation directory. After this, you can use `add package` and `upgrade cluster` to upgrade one or more clusters to the new version of the MySQL Cluster software.

Chapter 3 Using MySQL Cluster Manager

Table of Contents

3.1 <code>mcmd</code> , the MySQL Cluster Manager Agent	23
3.2 Starting and Stopping the MySQL Cluster Manager Agent	31
3.2.1 Starting and Stopping the Agent on Linux	32
3.2.2 Starting and Stopping the MySQL Cluster Manager Agent on Windows	33
3.3 Starting the MySQL Cluster Manager Client	35
3.4 Setting Up MySQL Clusters with MySQL Cluster Manager	38
3.4.1 Creating a MySQL Cluster with MySQL Cluster Manager	38
3.5 Importing MySQL Clusters into MySQL Cluster Manager	39
3.5.1 Importing a Cluster Into MySQL Cluster Manager: Basic Procedure	40
3.5.2 Importing a Cluster Into MySQL Cluster Manager: Example	41
3.6 MySQL Cluster Backup and Restore Using MySQL Cluster Manager	56
3.6.1 Requirements for Backup and Restore	56
3.6.2 Basic MySQL Cluster Backup and Restore Using MySQL Cluster Manager	56
3.7 Backing Up and Restoring MySQL Cluster Manager Agents	60
3.8 Setting Up MySQL Cluster Replication with MySQL Cluster Manager	62

This chapter discusses starting and stopping the MySQL Cluster Manager agent and client, and setting up, backing up, and restoring MySQL Clusters using the MySQL Cluster Manager.

3.1 `mcmd`, the MySQL Cluster Manager Agent

`mcmd` is the MySQL Cluster Manager agent program; invoking this executable starts the MySQL Cluster Manager Agent, to which you can connect using the `mcm` client (see [Section 3.3, “Starting the MySQL Cluster Manager Client”](#), and [Chapter 4, *MySQL Cluster Manager Client Commands*](#), for more information).

You can modify the behavior of the agent in a number of different ways by specifying one or more of the options discussed in this sections. Most of these options can be specified either on the command line or in the agent configuration file (normally `etc/mcmd.ini`). (Some exceptions include the `--defaults-file` and `--bootstrap` options, which, if used, must be specified on the command line, and which are mutually exclusive with one another.) For example, you can set the agent's cluster logging level to `warning` instead than the default `message` in either one of the following two ways:

- Include `--log-level=warning` on the command line when invoking `mcmd`.



Note

When specifying an agent configuration option on the command line, the name of the option is prefixed with two leading dash characters (`--`).

- Include the following line in the agent configuration file:

```
log-level=warning
```



Note

You can change the logging level at runtime using the `mcm` client `change log-level` command.

When used in the configuration file, the name of the option should not be prefixed with any other characters. Each option must be specified on a separate line. You can comment out all of a given line by inserting a leading hash character (`#`), like this:

```
#log-level=warning
```

You can also comment out part of a line in this way; any text following the `#` character is ignored, to the end of the current line.

The following table contains a summary of agent options that are read on startup by `mcmd`. More detailed information about each of these options, such as allowed range of values, can be found in the list following the table.

Table 3.1 MySQL Cluster Manager Agent (mcmd) Option Summary

Option Name	Description
<code>--agent-uuid</code>	Set the agent's UUID; needed only when running multiple agent processes on the same host
<code>--basedir</code>	Directory to use as prefix for relative paths in the configuration
<code>--bootstrap</code>	Bootstrap a default cluster on startup
<code>--daemon</code>	Run in daemon mode. The option applies only to Linux and other Unix-like platforms
<code>--defaults-file</code>	Configuration file to use
<code>--event-threads</code>	Number of event handler threads to use
<code>--help</code>	Show application options
<code>--help-all</code>	Show all options (application options and manager module options)
<code>--help-manager</code>	Show manager module options
<code>--keepalive</code>	Try to restart <code>mcmd</code> in the event of a crash. The option applies only to Linux and other Unix-like platforms
<code>--log-backtrace-on-crash</code>	Attempt to load debugger in case of a crash
<code>--log-file</code>	Name of the file to write the log to
<code>--log-level</code>	Set the <code>mcmd</code> logging level
<code>--log-use-syslog</code>	Log to syslog
<code>--manager-directory</code>	Directory used for manager data storage
<code>--manager-password</code>	Password for the <code>mcmd</code> user-account
<code>--manager-port</code>	Port for client to use when connecting to manager
<code>--manager-username</code>	Username for the <code>mcmd</code> user-account
<code>--max-open-files</code>	Maximum number of open files (<code>ulimit -n</code>)
<code>--pid-file</code>	Specify PID file (used if running as daemon)
<code>--plugin-dir</code>	Directory in which to look for plugins
<code>--plugins</code>	Comma-separated list of plugins to load; must include "manager"
<code>--verbose-shutdown</code>	Always log the exit code when shutting down
<code>--version</code>	Show the manager version
<code>--xcom-port</code>	Specify the XCOM port

MySQL Cluster Manager Agent (`mcmd`) Option Descriptions

The following list contains descriptions of each startup option available for use with `mcmd`, including allowed and default values. Options noted as boolean need only be specified in order to take effect; you should not try to set a value for these.

- `--agent-uuid=uuid`

Command-Line Format	<code>--agent-uuid=uuid</code>
Type	String
Default Value	[set internally]

Set a UUID for this agent. Normally this value is set automatically, and needs to be specified only when running more than one `mcmd` process on the same host.

- `--basedir=dir_name`

Command-Line Format	<code>--basedir=dir_name</code>
Type	Directory name
Default Value	.

Directory with path to use as prefix for relative paths in the configuration.

- `--bootstrap`

Command-Line Format	<code>--bootstrap</code>
---------------------	--------------------------

Start the agent with default configuration values, create a default one-machine cluster named `mycluster`, and start it. This option works only if no clusters have yet been created. This option is mutually exclusive with the `--defaults-file` option.

Currently, any data stored in the default cluster `mycluster` is not preserved between cluster restarts.

- `--daemon`

Command-Line Format	<code>--daemon</code>
Platform Specific	Linux

Run `mcmd` as a daemon.

- `--defaults-file=filename`

Command-Line Format	<code>--defaults-file=file_name</code>
Type	File name
Default Value	<code>etc/mcmd.ini</code>

Set the file from which to read configuration options. The default is `etc/mcmd.ini`. See [Section 2.4, “MySQL Cluster Manager Configuration File”](#), for more information.

- `--event-threads=#`

Command-Line Format	<code>--event-threads=#</code>
---------------------	--------------------------------

Type	Numeric
Default Value	1
Minimum Value	1
Maximum Value	[system dependent]

Number of event handler threads to use. The the default is 1, which is sufficient for most normal operations.

- `--help, -?`

Command-Line Format	<code>--help</code>
---------------------	---------------------

`mcmd` help output is divided into *Application* and *Manager* sections. When used with `mcmd`, `--help` causes the *Application* options to be shown, as shown here:

```
shell> mcmd --help
Usage:
  mcmd [OPTION...] - MySQL Cluster Manager

Help Options:
  -?, --help                Show help options
  --help-all               Show all help options
  --help-manager            Show options for the manager-module

Application Options:
  -V, --version              Show version
  --defaults-file=<file>    configuration file
  --verbose-shutdown        Always log the exit code when shutting down
  --daemon                  Start in daemon-mode
  --basedir=<absolute path> Base directory to prepend to relative paths in the config
  --pid-file=<file>         PID file in case we are started as daemon
  --plugin-dir=<path>       Path to the plugins
  --plugins=<name>          Plugins to load
  --log-level=<string>      Log all messages of level ... or higher
  --log-file=<file>         Log all messages in a file
  --log-use-syslog          Log all messages to syslog
  --log-backtrace-on-crash  Try to invoke debugger on crash
  --keepalive               Try to restart mcmd if it crashed
  --max-open-files          Maximum number of open files (ulimit -n)
  --event-threads           Number of event-handling threads (default: 1)
```

- `--help-all`

Command-Line Format	<code>--help-all</code>
---------------------	-------------------------

`mcmd` help output is divided into *Application* and *Manager* sections. When used with `--help-all`, `mcmd` displays both the *Application* and the *Manager* options, like this:

```
> mcmd --help-all
Usage:
  mcmd [OPTION...] - MySQL Cluster Manager

Help Options:
  -?, --help                Show help options
  --help-all               Show all help options
  --help-manager            Show options for the manager-module

manager-module
  --manager-port=<clientport> Port to manage the cluster (default: 1862)
  --xcom-port=<xcomport>       Xcom port (default: 18620)
  --manager-username=<username> Username to manage the cluster (default: mcmd)
```


<code>--manager-password=<password></code>	Password for the manager user-account (default: super)
<code>--bootstrap</code>	Bootstrap a default cluster on initial startup
<code>--manager-directory=<directory></code>	Path to managers config information
Application Options:	
<code>-V, --version</code>	Show version
<code>--defaults-file=<file></code>	configuration file
<code>--verbose-shutdown</code>	Always log the exit code when shutting down
<code>--daemon</code>	Start in daemon-mode
<code>--basedir=<absolute path></code>	Base directory to prepend to relative paths in the config
<code>--pid-file=<file></code>	PID file in case we are started as daemon
<code>--plugin-dir=<path></code>	Path to the plugins
<code>--plugins=<name></code>	Plugins to load
<code>--log-level=<string></code>	Log all messages of level ... or higher
<code>--log-file=<file></code>	Log all messages in a file
<code>--log-use-syslog</code>	Log all messages to syslog
<code>--log-backtrace-on-crash</code>	Try to invoke debugger on crash
<code>--keepalive</code>	Try to restart mcmd if it crashed
<code>--max-open-files</code>	Maximum number of open files (ulimit -n)
<code>--event-threads</code>	Number of event-handling threads (default: 1)

- `--help-manager`

Command-Line Format	<code>--help-manager</code>
---------------------	-----------------------------

mcmd help output is divided into *Application* and *Manager* sections. When used with `--help-manager`, mcmd displays the *Manager* options, like this:

```
shell> mcmd --help-manager
Usage:
  mcmd [OPTION...] - MySQL Cluster Manager

manager-module
  --manager-port=<clientport>      Port to manage the cluster (default: 1862)
  --xcom-port=<xcomport>           Xcom port (default: 18620)
  --manager-username=<username>    Username to manage the cluster (default: mcmd)
  --manager-password=<password>    Password for the manager user-account (default: super)
  --bootstrap                      Bootstrap a default cluster on initial startup
  --manager-directory=<directory>  Path to managers config information
```

- `--keepalive`

Command-Line Format	<code>--keepalive</code>
Platform Specific	Linux

Use this option to cause mcmd to attempt to restart in the event of a crash.

- `--log-backtrace-on-crash`

Command-Line Format	<code>--log-backtrace-on-crash</code>
---------------------	---------------------------------------

Attempt to load the debugger in the event of a crash. Not normally used in production.

- `--log-file=filename`

Command-Line Format	<code>--log-file=file</code>
Type	File name

Default Value	mcmd.log
---------------	----------

Set the name of the file to write the log to. The default is `mcmd.log` in the installation directory. On Linux and other Unix-like platforms, you can use a relative path; this is in relation to the MySQL Cluster Manager installation directory, and not to the `bin` or `etc` subdirectory. On Windows, you must use an absolute path, and it cannot contain any spaces; in addition, you must replace any backslash (`\`) characters in the path with forward slashes (`/`).

- `--log-level=level`

Command-Line Format	<code>--log-level=level</code>
Type	Enumeration
Default Value	<code>message</code>
Valid Values	<code>critical</code> <code>error</code> <code>warning</code> <code>message</code> <code>info</code> <code>debug</code>

Sets the `mcmd` log severity level. Possible values for this option and their descriptions are listed in [Table 3.2, “MySQL Cluster Manager Agent Log Levels”](#) in descending level of severity. When the option is set to a certain severity level, all events of that or higher levels are logged. `message` is the default log level, and is the recommended setting for a production environment; running on a more severe log level produces fewer messages and makes it harder to trace a problem when it occurs.

Table 3.2 MySQL Cluster Manager Agent Log Levels

Level of Severity	Description
<code>critical</code>	Conditions that should be corrected immediately, such as a corrupted MySQL Cluster Manager data repository
<code>error</code>	Conditions that should be corrected, such as configuration errors
<code>warning</code>	Conditions that do not fail executions, but may require user attention
<code>message</code>	Messages on main events of the site and from command execution
<code>info</code>	Informational messages to provide users with some execution details
<code>debug</code>	Debugging messages that give execution details useful for developers. This causes large log files if used over a long period of time.

You can also change the `mcmd` logging level at runtime using the `change log-level` command with the `mcm` client. While the setting of the `--log-level` option is applied only to the host whose `mcmd`

agent uses the option (either at the command line or in a configuration file), the `change log-level` client command can be used to apply the logging level to an entire management site or to specific hosts.

- `--log-use-syslog`

Command-Line Format	<code>--log-use-syslog</code>
---------------------	-------------------------------

Write logging output to syslog.

- `--manager-directory=dir_name`

Command-Line Format	<code>--manager-directory=dir</code>
Type	Directory name
Default Value	<code>../mcm_data</code> (relative to the installation directory of MySQL Cluster Manager)

Set the location of the agent repository, which contains collections of MySQL Cluster Manager data files and MySQL Cluster configuration and data files. The value must be a valid absolute path. On Linux, if the directory does not exist, it is created; on Windows, the directory must be created if it does not exist. additionally on Windows, the path may not contain any spaces or backslash (\) characters; backslashes must be replaced with forward slashes (/).

The default location is `../mcm_data` (relative to the installation directory of MySQL Cluster Manager). If you change the default, you should use a standard location external to the MySQL Cluster Manager installation directory, such as `/var/opt/mcm` on Linux.

In addition to the MySQL Cluster Manager data files, the `manager-directory` also contains a `rep` directory in which MySQL Cluster data files for each MySQL Cluster under MySQL Cluster Manager control are kept. Normally, there is no need to interact with these directories beyond specifying the location of the `manager-directory` in the agent configuration file (`mcmd.ini`).

However, in the event that an agent reaches an inconsistent state, it is possible to delete the contents of the `rep` directory, in which case the agent attempts to recover its repository from another agent. *In such cases, you must also delete the `repchksum` file and the `high_water_mark` file from the `manager-directory`.* Otherwise, the agent reads these files and raises errors due to the now-empty `rep` directory.

- `--manager-password=password`

Command-Line Format	<code>--manager-password=password</code>
Type	String
Default Value	<code>super</code>

Set a password to be used for the manager agent user account. The default is `super`.

Using this option together with `manager-username` causes the creation of a MySQL user account, having the username and password specified using these two options. *This account is created with all privileges on the MySQL server including the granting of privileges.* In other words, it is created as if you had executed `GRANT ALL PRIVILEGES ON *.* ... WITH GRANT OPTION` in the `mysql` client.

- `--manager-port=#`

Command-Line Format	<code>--manager-port=port</code>
---------------------	----------------------------------

Type	Numeric
Default Value	<code>localhost:1862</code>

Specify the port used by MySQL Cluster Manager client connections. Any valid TC/IP port number can be used. Normally, there is no need to change it from the default value (1862).

Previously, this option could optionally take a host name in addition to the port number, but in MySQL Cluster Manager 1.1.1 and later the host name is no longer accepted.

- `--manager-username=user_name`

Command-Line Format	<code>--manager-username=name</code>
Type	String
Default Value	<code>mcmd</code>

Set a user name for the MySQL account to be used by the MySQL Cluster Manager agent. The default is `mcmd`.

When used together with `manager-password`, this option also causes the creation of a new MySQL user account, having the user name and password specified using these two options. *This account is created with all privileges on the MySQL server including the granting of privileges.* In other words, it is created as if you had executed `GRANT ALL PRIVILEGES ON *.* ... WITH GRANT OPTION` in the `mysql` client. The existing MySQL `root` account is not altered in such cases, and the default `test` database is preserved.

- `--max-open-files=#`

Command-Line Format	<code>--max-open-files=#</code>
Type	Numeric
Default Value	<code>1</code>
Minimum Value	<code>1</code>
Maximum Value	<code>[system dependent]</code>

Set the maximum number of open files (as with `ulimit -n`).

- `--pid-file=file`

Command-Line Format	<code>--pid-file=file_name</code>
Type	File name
Default Value	<code>mcmd.pid</code>

Set the name and path to a process ID (`.pid`) file. Not normally used or needed. This option is not supported on Windows systems.

- `--plugin-dir`

Command-Line Format	<code>--plugin-dir=dir_name</code>
Type	Directory name

Default Value	<code>lib/mcmd</code>
---------------	-----------------------

Set the directory to search for plugins. The default is `lib/mcmd`, in the MySQL Cluster Manager installation directory; normally there is no need to change this.

- `--plugins`

Command-Line Format	<code>--plugins=list</code>
Type	Directory name
Default Value	<code>manager</code>

Specify a list of plugins to be loaded on startup. To enable MySQL Cluster Manager, this list must include `manager` (the default value). Please be aware that we currently do not test MySQL Cluster Manager with any values for `plugins` other than `manager`. Therefore, we recommend using the default value in a production setting.

- `--verbose-shutdown`

Command-Line Format	<code>--verbose-shutdown</code>
---------------------	---------------------------------

Force `mcmd` to log the exit code whenever shutting down, regardless of the reason.

- `--version, -V`

Command-Line Format	<code>--version</code>
---------------------	------------------------

Display version information and exit. Output may vary according to the MySQL Cluster Manager software version, operating platform, and versions of libraries used on your system, but should closely resemble what is shown here, with the first line of output containing the MySQL Cluster Manager release number (emphasized text):

```
shell> mcmd -V
MySQL Cluster Manager 1.3.6 (64bit)
  chassis: mysql-proxy 0.8.3
  glib2: 2.16.6
  libevent: 1.4.13-stable
-- modules
  manager: 1.3.6
```

- `--xcom-port`

Command-Line Format	<code>--xcom-port=port</code>
Type	Numeric
Default Value	<code>18620</code>

Allows you to specify the XCOM port. The default is 18620.

3.2 Starting and Stopping the MySQL Cluster Manager Agent

Before you can start using MySQL Cluster Manager to create and manage a MySQL Cluster, the MySQL Cluster Manager agent must be started on each computer that is intended to host one or more nodes in the MySQL Cluster to be managed.

The MySQL Cluster Manager agent employs a MySQL user account for administrative access to `mysqld` processes. It is possible, but not a requirement, to change the default user name, the default password

used for this account, or both. For more information, see [Section 2.3.3, “Setting the MySQL Cluster Manager Agent User Name and Password”](#).

3.2.1 Starting and Stopping the Agent on Linux

To start the MySQL Cluster Manager agent on a given host running a Linux or similar operating system, you should run `mcmd`, found in the `bin` directory within the manager installation directory on that host. Typical options used with `mcmd` are shown here:

```
mcmd [--defaults-file | --bootstrap] [--log-file] [--log-level]
```

See [Section 3.1, “mcmd, the MySQL Cluster Manager Agent”](#), for information about additional options that can be used when invoking `mcmd` from the command line, or in a configuration file.

`mcmd` normally runs in the foreground. If you wish, you can use your platform's usual mechanism for backgrounding a process. On a Linux system, you can do this by appending an ampersand character (`&`), like this (not including any options that might be required):

```
shell> ./bin/mcmd &
```

By default, the agent assumes that the agent configuration file is `etc/mcmd.ini`, in the MySQL Cluster Manager installation directory. You can tell the agent to use a different configuration file by passing the path to this file to the `--defaults-file` option, as shown here:

```
shell> ./bin/mcmd --defaults-file=/home/mcm/mcm-agent.conf
```

The `--bootstrap` option causes the agent to start with default configuration values, create a default one-machine cluster named `mycluster`, and start it. This option works only if no cluster has yet created, and is mutually exclusive with the `--defaults-file` option. Currently, any data stored in the default cluster `mycluster` is not preserved between cluster restarts; this is a known issue which we may address in a future release of MySQL Cluster Manager.

The use of the `--bootstrap` option with `mcmd` is shown here on a system having the host name `torsk`, where MySQL Cluster Manager has been installed to `/home/jon/mcm`:

```
shell> ./mcmd --bootstrap
MySQL Cluster Manager 1.3.6 started
Connect to MySQL Cluster Manager by running "/home/jon/mcm/bin/mcm" -a torsk:1862
Configuring default cluster 'mycluster'...
Starting default cluster 'mycluster'...
Cluster 'mycluster' started successfully
      ndb_mgmd      torsk:1186
      ndbd         torsk
      ndbd         torsk
      mysqld       torsk:3306
      mysqld       torsk:3307
      ndbapi       *
Connect to the database by running "/home/jon/mcm/cluster/bin/mysql" -h torsk -P 3306 -u root
```

You can then connect to the agent using the `mcm` client (see [Section 3.3, “Starting the MySQL Cluster Manager Client”](#)), and to either of the MySQL Servers running on ports 3306 and 3307 using `mysql` or another MySQL client application.

The `--log-file` option allows you to override the default location for the agent log file (normally `mcmd.log`, in the MySQL Cluster Manager installation directory).

You can use `--log-level` option to override the `log-level` set in the agent configuration file.

See [Section 3.1, “mcmd, the MySQL Cluster Manager Agent”](#), for more information about options that can be used with `mcmd`.

The MySQL Cluster Manager agent must be started on each host in the MySQL Cluster to be managed.

To stop one or more instances of the MySQL Cluster Manager agent, use the `stop agents` command in the MySQL Cluster Manager client. If the client is unavailable, you can stop each agent process using the system's standard method for doing so, such as `^C` or `kill`.

You can also set the agent up as a daemon or service on Linux and other Unix-like systems. (See [Section 2.3.1, "Installing MySQL Cluster Manager on Unix Platforms"](#).) If you also want data node failed processes from a running MySQL Cluster to be started when the agent fails and restarts in such cases, you must make sure that `StopOnError` is set to 0 on each data node (and not to 1, the default).

3.2.2 Starting and Stopping the MySQL Cluster Manager Agent on Windows

To start the MySQL Cluster Manager agent manually on a Windows host, you should invoke `mcmd.exe`, found in the `bin` directory under the manager installation directory on that host. By default, the agent uses `etc/mcmd.ini` in the MySQL Cluster Manager installation directory as its configuration file; this can be overridden by passing the desired file's location as the value of the `--defaults-file` option.

Typical options for `mcmd` are shown here:

```
mcmd[.exe] [--defaults-file | --bootstrap] [--log-file] [--log-level]
```

For information about additional options that can be used with `mcmd` on the command line or in an option file, see [Section 3.1, "mcmd, the MySQL Cluster Manager Agent"](#).

By default, the agent assumes that the agent configuration file is `etc/mcmd.ini`, in the MySQL Cluster Manager installation directory. You can tell the agent to use a different configuration file by passing the path to this file to the `--defaults-file` option, as shown here:

```
C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.1.4\bin>
mcmd --defaults-file="C:\Program Files (x86)\MySQL\MySQL Cluster
Manager 1.3.6\etc\mcmd.ini"
```

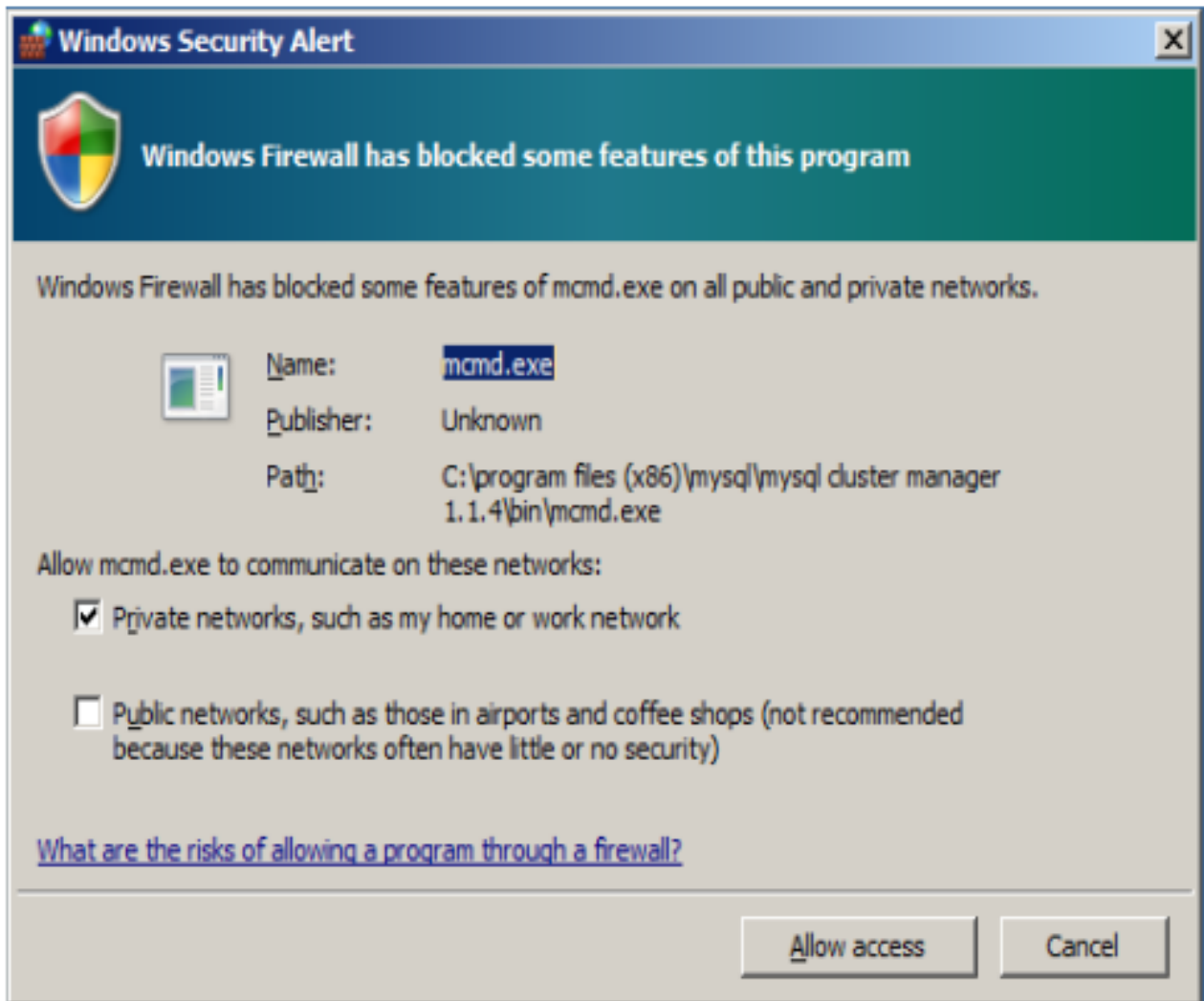
The `--bootstrap` option causes the agent to start with default configuration values, create a default one-machine cluster named `mycluster`, and start it. The use of this option with `mcmd` is shown here on a system having the host name `torsk`, where MySQL Cluster Manager has been installed to the default location:

```
C:\Program Files (x86)\MySQL\MySQL Cluster Manager 1.3.6\bin>mcmd --bootstrap
MySQL Cluster Manager 1.3.6 started
Connect to MySQL Cluster Manager by running "C:\Program Files (x86)\MySQL\MySQL
Cluster Manager 1.3.6\bin\mcm" -a TORSK:1862
Configuring default cluster 'mycluster'...
Starting default cluster 'mycluster'...
Cluster 'mycluster' started successfully
      ndb_mgmd      TORSK:1186
      ndbd          TORSK
      ndbd          TORSK
      mysqld        TORSK:3306
      mysqld        TORSK:3307
      ndbapi        *
Connect to the database by running "C:\Program Files (x86)\MySQL\MySQL Cluster
Manager 1.3.6\cluster\bin\mysql" -h TORSK -P 3306 -u root
```

You can then connect to the agent using the `mcm` client (see [Section 3.3, "Starting the MySQL Cluster Manager Client"](#)), and to either of the MySQL Servers running on ports 3306 and 3307 using `mysql` or another MySQL client application.

When starting the MySQL Cluster Manager agent for the first time, you may see one or more Windows **Security Alert** dialogs, such as the one shown here:

Figure 3.1 Starting the MySQL Cluster Manager Agent on Windows: Security Alert



You should grant permission to connect to private networks for any of the programs `mcmd.exe`, `ndb_mgmd.exe`, `ndbd.exe`, `ndbmtbd.exe`, or `mysqld.exe`. To do so, check the **Private Networks...** box and then click the **Allow access** button. It is generally not necessary to grant MySQL Cluster Manager or MySQL Cluster access to public networks such as the Internet.



Note

The `--defaults-file` and `--bootstrap` options are mutually exclusive.

The `--log-file` option allows you to override the default location for the agent log file (normally `mcmd.log`, in the MySQL Cluster Manager installation directory).

You can use `--log-level` option to override the `log-level` set in the agent configuration file.

See [Section 3.1, “mcmd, the MySQL Cluster Manager Agent”](#), for more information about options that can be used with `mcmd`.

The MySQL Cluster Manager agent must be started on each host in the MySQL Cluster to be managed.

It is possible to install MySQL Cluster Manager as a Windows service, so that it is started automatically each time Windows starts. See [Section 2.3.2.1, “Installing the MySQL Cluster Manager Agent as a Windows Service”](#).

To stop one or more instances of the MySQL Cluster Manager agent, use the `stop agents` command in the MySQL Cluster Manager client. You can also stop an agent process using the Windows Task Manager. In addition, if you have installed MySQL Cluster Manager as a Windows service, you can stop (and start) the agent using the Windows Service Manager, `CTRL-C`, or the appropriate `SC STOP` (or `SC START`) or `NET STOP` (or `NET START`) command. See [Starting and stopping the MySQL Cluster Manager agent Windows service](#), for more information about each of these options.

3.3 Starting the MySQL Cluster Manager Client

This section covers starting the MySQL Cluster Manager client and connecting to the MySQL Cluster Manager agent.

MySQL Cluster Manager 1.3.6 includes a command-line client `mcm`, located in the installation `bin` directory. `mcm` can be invoked with any one of the options shown in the following table:

Table 3.3 mcm command line options

Long form	Short form	Description
<code>--help</code>	<code>-?</code>	Display <code>mcm</code> client options
<code>--version</code>	<code>-V</code>	Shows MySQL Cluster Manager agent/client version.
<code>—</code>	<code>-W</code>	Shows MySQL Cluster Manager agent/client version, with version of <code>mysql</code> used by <code>mcm</code> .
<code>--address</code>	<code>-a</code>	Host and optional port to use when connecting to <code>mcmd</code> , in <code>host[:port]</code> format; default is <code>127.0.0.1:1862</code> .
<code>--mysql-help</code>	<code>-I</code>	Show help for <code>mysql</code> client (see following).

The client/server protocol used by MySQL Cluster Manager is platform-independent. You can connect to any MySQL Cluster Manager agent with an `mcm` client on any platform where it is available. This means, for example, that you can use an `mcm` client on Microsoft Windows to connect to a MySQL Cluster Manager agent that is running on a Linux host.

`mcm` actually acts as a wrapper for the `mysql` client that is included with the bundled MySQL Cluster distribution. Invoking `mcm` with no options specified is equivalent to the following:

```
shell> mysql -umcmd -psuper -h 127.0.0.1 -P 1862 --prompt="mcm>"
```

(These `-u` and `-p` options and values are hard-coded and cannot be changed.) This means that you can use the `mysql` client to run MySQL Cluster Manager client sessions on platforms where `mcm` itself (or even `mcmd`) is not available. For more information, see [Connecting to the agent using the mysql client](#).

If you experience problems starting an MySQL Cluster Manager client session because the client fails to connect, see [Can't connect to \[local\] MySQL server](#), for some reasons why this might occur, as well as suggestions for some possible solutions.

To end a client session, use the `exit` or `quit` command (short form: `\q`). Neither of these commands requires a separator or terminator character.

For more information, see [Chapter 4, MySQL Cluster Manager Client Commands](#).

Connecting to the agent with the `mcm` client. You can connect to the MySQL Cluster Manager agent by invoking `mcm` (or, on Windows, `mcm.exe`). You may also need to specify a hostname, port number, or both, using the following command-line options:

- `--host=hostname` or `-h[]hostname`

This option takes the name or IP address of the host to connect to. The default is `localhost` (which may not be recognized on all platforms when starting a `mcm` client session even if it works for starting `mysql` client sessions).

You should keep in mind that the `mcm` client does not perform host name resolution; any name resolution information comes from the operating system on the host where the client is run. For this reason, it is usually best to use a numeric IP address rather than a hostname for this option.

- `--port=portnumber` or `-P[]portnumber`

This option specifies the TCP/IP port for the client to use. This must be the same port that is used by the MySQL Cluster Manager agent. As mentioned elsewhere, if no agent port is specified in the MySQL Cluster Manager agent configuration file (`mcmd.ini`), the default number of the port used by the MySQL Cluster Manager agent is 1862, which is also used by default by `mcm`.

`mcm` accepts additional `mysql` client options, some of which may possibly be of use for MySQL Cluster Manager client sessions. For example, the `--pager` option might prove helpful when the output of `get` contains too many rows to fit in a single screen. The `--prompt` option can be used to provide a distinctive prompt to help avoid confusion between multiple client sessions. However, options not shown in the current manual have not been extensively tested with `mcm` and so cannot be guaranteed to work correctly (or even at all). See [mysql Client Options](#), for a complete listing and descriptions of all `mysql` client options.



Note

Like the `mysql` client, `mcm` also supports `\G` as a statement terminator which causes the output to be formatted vertically. This can be helpful when using a terminal whose width is restricted to some number of (typically 80) characters. See [Chapter 4, MySQL Cluster Manager Client Commands](#), for examples.

Connecting to the agent using the `mysql` client. As mentioned previously, `mcm` actually serves as a wrapper for the `mysql` client. In fact, a `mysql` client from any recent MySQL distribution (MySQL 5.1 or later) should work without any issues for connecting to `mcmd`. In addition, since the client/server protocol used by MySQL Cluster Manager is platform-independent, you can use a `mysql` client on any platform supported by MySQL. (This means, for example, that you can use a `mysql` client on Microsoft Windows to connect to a MySQL Cluster Manager agent that is running on a Linux host.) Connecting to the MySQL Cluster Manager agent using the `mysql` client is accomplished by invoking `mysql` and specifying a hostname, port number, username and password, using the following command-line options:

- `--host=hostname` or `-h[]hostname`

This option takes the name or IP address of the host to connect to. The default is `localhost`. Like the `mcm` client, the `mysql` client does not perform host name resolution, and relies on the host operating system for this task. For this reason, it is usually best to use a numeric IP address rather than a hostname for this option.

- `--port=portnumber` or `-P[]portnumber`

This option specifies the TCP/IP port for the client to use. This must be the same port that is used by the MySQL Cluster Manager agent. Although the default number of the port used by the MySQL Cluster Manager agent is 1862 (which is also used by default by `mcm`), *this default value is not known to the `mysql` client*, which uses port 3306 (the default port for the MySQL server) if this option is not specified when `mysql` is invoked.

Thus, you *must* use the `--port` or `-P` option to connect to the MySQL Cluster Manager agent using the `mysql` client, *even if the agent process is using the MySQL Cluster Manager default port*, and even if the agent process is running on the same host as the `mysql` client. Unless the correct agent port number is supplied to it on startup, `mysql` is unable to connect to the agent.

- `--user=username` or `-u[]username`

Specifies the username for the user trying to connect. Currently, the only user permitted to connect is “mcmd”; this is hard-coded into the agent software and cannot be altered by any user. By default, the `mysql` client tries to use the name of the current system user on Unix systems and “ODBC” on Windows, so you *must* supply this option and the username “mcmd” when trying to access the MySQL Cluster Manager agent with the `mysql` client; otherwise, `mysql` cannot connect to the agent.

- `--password=[password]` or `-p[password]`

Specifies the password for the user trying to connect. If you use the short option form (`-p`), you *must not* leave a space between this option and the password. If you omit the `password` value following the `--password` or `-p` option on the command line, the `mysql` client prompts you for one.

Specifying a password on the command line should be considered insecure. It is preferable that you either omit the password when invoking the client, then supply it when prompted, or put the password in a startup script or configuration file.

Currently, the password is hard-coded as “super”, and cannot be changed or overridden by MySQL Cluster Manager users. Therefore, if you do not include the `--password` or `-p` option when invoking `mysql`, it cannot connect to the agent.

In addition, you can use the `--prompt` option to set the `mysql` client's prompt. This is recommended, since allowing the default prompt (`mysql>`) to be used could lead to confusion between a MySQL Cluster Manager client session and a MySQL client session.

Thus, you can connect to a MySQL Cluster Manager agent by invoking the `mysql` client on the same machine from the system shell in a manner similar to what is shown here.

```
shell> mysql -h127.0.0.1 -P1862 -umcmd -p --prompt='mcm> '
```

For convenience, on systems where `mcm` itself is not available, you might even want to put this invocation in a startup script. On a Linux or similar system, this script might be named `mcm-client.sh`, with contents similar to what is shown here:

```
#!/bin/sh
/usr/local/mysql/bin/mysql -h127.0.0.1 -P1862 -umcmd -p --prompt='mcm> '
```

In this case, you could then start up a MySQL Cluster Manager client session using something like this in the system shell:

```
shell> ./mcm-client
```

On Windows, you can create a batch file with a name such as `mcm-client.bat` containing something like this:

```
C:\mysql\bin\mysql.exe -umcmd -psuper -h localhost -P 1862 --prompt="mcm> "
```

(Adjust the path to the `mysql.exe` client executable as necessary to match its location on your system.)

If you save this file to a convenient location such as the Windows desktop, you can start a MySQL Cluster Manager client session merely by double-clicking the corresponding file icon on the desktop (or in Windows Explorer); the client session opens in a new `cmd.exe` (DOS) window.

3.4 Setting Up MySQL Clusters with MySQL Cluster Manager

This section provides basic information about setting up a new MySQL Cluster with MySQL Cluster Manager. It also supplies guidance on migration of an existing MySQL Cluster to MySQL Cluster Manager.

For more information about obtaining and installing the MySQL Cluster Manager agent and client software, see [Chapter 2, MySQL Cluster Manager Installation, Configuration, Cluster Setup](#).

See [Chapter 4, MySQL Cluster Manager Client Commands](#), for detailed information on the MySQL Cluster Manager client commands shown in this chapter.

3.4.1 Creating a MySQL Cluster with MySQL Cluster Manager

In this section, we discuss the procedure for using MySQL Cluster Manager to create and start a new MySQL Cluster. We assume that you have already obtained the MySQL Cluster Manager and MySQL Cluster software, and that you are already familiar with installing MySQL Cluster Manager (see [Chapter 2, MySQL Cluster Manager Installation, Configuration, Cluster Setup](#)).

MySQL Cluster Manager 1.3.0 and later also supports importing existing, standalone MySQL Clusters; for more information, see [Section 3.5, "Importing MySQL Clusters into MySQL Cluster Manager"](#).

We also assume that you have identified the hosts on which you plan to run the cluster and have decided on the types and distributions of the different types of nodes among these hosts, as well as basic configuration requirements based on these factors and the hardware characteristics of the host machines.



Note

You can create and start a MySQL Cluster on a single host for testing or similar purposes, simply by invoking `mcmd` with the `--bootstrap` option. See [Section 3.2, "Starting and Stopping the MySQL Cluster Manager Agent"](#).

Creating a new cluster consists of the following tasks:

- **MySQL Cluster Manager agent installation and startup.** Install the MySQL Cluster Manager software distribution, make any necessary edits of the agent configuration files, and start the agent processes as explained in [Chapter 2, MySQL Cluster Manager Installation, Configuration, Cluster Setup](#). Agent processes must be running on all cluster hosts before you can create a cluster. This means that you need to place a complete copy of the MySQL Cluster Manager software distribution on every host. The MySQL Cluster Manager software does not have to be in a specific location, or even the same location on all hosts, but it must be present; you cannot manage any cluster processes hosted on a computer where `mcmd` is not also running.
- **MySQL Cluster Manager client session startup.** Starting the MySQL Cluster Manager client and connect to the MySQL Cluster Manager agent. You can connect to an agent process running on any of the cluster hosts, using the `mcm` client on any computer that can establish a network connection to the desired host. See [Section 3.3, "Starting the MySQL Cluster Manager Client"](#), for details.

On systems where `mcm` is not available, you can use the `mysql` client for this purpose. See [Connecting to the agent using the `mysql` client](#).

- **MySQL Cluster software deployment.** The simplest and easiest way to do this is to copy the complete MySQL Cluster distribution to the same location on every host in the cluster. (If you have installed MySQL Cluster Manager 1.3.6 on each host, the MySQL Cluster NDB 7.2.4 distribution is already included, in `mcm_installation_dir/cluster`.) If you do not use the same location on every host, be sure to note it for each host. Do not yet start any MySQL Cluster processes or edit any configuration files; when creating a new cluster, MySQL Cluster Manager takes care of these tasks automatically.

On Windows hosts, you should *not* install as services any of the MySQL Cluster node process programs, including `ndb_mgmd.exe`, `ndbd.exe`, `ndbmttd.exe`, and `mysqld.exe`. MySQL Cluster Manager manages MySQL Cluster processes independently of the Windows Service Manager and does not interact with the Service Manager or any Windows services when doing so.

**Note**

You can actually perform this step at any time up to the point where the software package is registered (using `add package`). However, we recommend that you have all required software—including the MySQL Cluster software—in place before executing any MySQL Cluster Manager client commands.

- **Management site definition.** Using the `create site` command in the MySQL Cluster Manager client, define a MySQL Cluster Manager management site—that is, the set of hosts to be managed. This command provides a name for the site, and must reference all hosts in the cluster. [Section 4.2.6, “The create site Command”](#), provides syntax and other information about this command. To verify that the site was created correctly, use the MySQL Cluster Manager client commands `list sites` and `list hosts`.
- **MySQL Cluster software package registration.** In this step, you provide the location of the MySQL Cluster software on all hosts in the cluster using one or more `add package` commands. To verify that the package was created correctly, use the `list packages` and `list processes` commands.
- **Cluster definition.** Execute a `create cluster` command to define the set of MySQL Cluster nodes (processes) and hosts on which each cluster process runs, making up a the MySQL Cluster. This command also uses the name of the package registered in the previous step so that MySQL Cluster Manager knows the location of the binary running each cluster process. You can use the `list clusters` and `list processes` commands to determine whether the cluster has been defined as desired.

If you wish to use SQL node connection pooling, see [Setup for mysqld connection pooling](#) before creating the cluster.

- **Initial configuration.** Perform any configuration of the cluster that is required or desired prior to starting it. You can set values for MySQL Cluster Manager configuration attributes (MySQL Cluster parameters and MySQL Server options) using the MySQL Cluster Manager client `set` command. You do not need to edit any configuration files directly—in fact, you should *not* do so. Keep in mind that certain attributes are read-only, and that some others cannot be reset after the cluster has been started for the first time. You can use the `get` command to verify that attributes have been set to the correct values.
- **Cluster startup.** Once you have completed the previous steps, including necessary or desired initial configuration, you are ready to start the cluster. The `start cluster` command starts all cluster processes in the correct order. You can verify that the cluster has started and is running normally after this command has completed, using the MySQL Cluster Manager client command `show status`. At this point, the cluster is ready for use by MySQL Cluster applications.

3.5 Importing MySQL Clusters into MySQL Cluster Manager

It is possible to bring a “wild” MySQL Cluster—that is, a cluster not created using MySQL Cluster Manager—under the control of MySQL Cluster Manager. The following sections provide an outline of the procedure required to import such a cluster into MySQL Cluster Manager, followed by a more detailed example.

3.5.1 Importing a Cluster Into MySQL Cluster Manager: Basic Procedure

The importation process consists generally of following the steps listed here:

1. Create and configure in MySQL Cluster Manager a “target” cluster whose configuration matches that of the “wild” cluster.
2. Prepare the “wild” cluster for migration.
3. Verify PID files for cluster processes.
4. Perform a test run, and then execute the `import cluster` command.

This expanded listing breaks down each of the tasks just mentioned into smaller steps; an example with more detail is also provided following the listing.

1. *Create and configure “target” cluster under MySQL Cluster Manager control*
 - a. Install MySQL Cluster Manager and start `mcmd` on all hosts; see [Section 2.3, “MySQL Cluster Manager Installation”](#), for more information.
 - b. Create a MySQL Cluster Manager site encompassing these hosts, using the `create site` command.
 - c. Add a MySQL Cluster Manager package referencing the MySQL Cluster binaries, using the `add package` command. Use this command's `--basedir` option to point to the correct location.
 - d. Create the target cluster using the `create cluster` command, including the same processes and hosts used by the wild cluster. Use the command's `--import` option to specify that the cluster is a target for import.

If the wild cluster adheres to the recommendation for node ID assignments given in the description for the `create cluster` command (that is, having node ID 1 to 48 assigned to data nodes, and 49 and above assigned to other node types), you need not specify the node IDs for the processes in the `create cluster` command.

Also, this step may be split into a `create cluster` command followed by one or more `add process` commands (see an example of such splitting in the description for the `add process` command).

- e. *MySQL Cluster Manager 1.3.1 and later.* Use `import config` to copy the wild cluster's configuration data into the target cluster. Use this command's `--dryrun` option (short form: `-y`) to perform a test run that merely logs the configuration information that the command copies when it is executed without the option.

If any `ndb_mgmd` or `mysqld` processes in the wild cluster are running on ports other than the default, you must perform `set` commands to assign the correct port numbers for these in the target cluster. When all such processes are running on the correct ports, you can execute `import config` (without the `--dryrun` option) to copy the wild cluster's configuration data. Following this step, you should check the log as well as the configuration of the target cluster to ensure that all configuration attribute values were copied correctly and with the correct scope. Correct any inconsistencies with the wild cluster's configuration using the appropriate `set` commands.

MySQL Cluster Manager 1.3.0: Since `import config` is not supported prior to the MySQL Cluster Manager 1.3.1 release, you must copy the wild cluster's configuration information to the target cluster manually, issuing `set` commands in the `mcm` client that duplicate the wild cluster's configuration in the target cluster, as discussed in the paragraphs immediately following.

MySQL Cluster global configuration data is stored in a file on the management node host which is usually (but not always) named `config.ini`. This global configuration file uses INI format which makes it simple to read or parse. For more information about this file, see [NDB Cluster Configuration Files](#), and [NDB Cluster Configuration: Basic Example](#)

In addition, each `mysqld` process (SQL node) has its own configuration data in the form of system variables which are specific to that `mysqld`, and many of which can be changed at runtime. You can check their values using the SQL `SHOW VARIABLES` statement, and execute appropriate `set` commands for values differing from their defaults.

2. *Prepare the “wild” cluster for migration*
 - a. Create a MySQL user named `mcmd` on each SQL node, and grant root privileges to this user.
 - b. Kill each data node angel process using your system's facility for doing so. Do not kill any non-angel data node daemons.
 - c. Kill and restart each management node process. When restarting `ndb_mgmd`, be sure to do so with the configuration cache disabled. Since the configuration cache is enabled by default, you must start the management node with `--config-cache=false` to deactivate it.
 - d. Any cluster processes that are under the control of the system's boot-time process management facility, such as `/etc/init.d` on Linux systems or the Services Manager on Windows platforms, should be removed from its control.
 - e. It is highly recommended that you take a complete backup of the “wild” cluster before proceeding any further, using the `ndb_mgm` client. For more information, see [Using The NDB Cluster Management Client to Create a Backup](#).

3. *Verify cluster process PID files.*
 - a. Verify that each process in the “wild” cluster has a valid PID file.
 - b. If a given process does not have a valid PID file, you must create one for it.

See [Section 3.5.2.3, “Verify All Cluster Process PID Files”](#), for a more detailed explanation and examples.

4. *Test and perform migration of “wild” cluster.*
 - a. Perform a test run of the proposed migration using `import cluster` with the `--dryrun` option, which causes MySQL Cluster Manager to check for errors, but not actually migrate any processes or data.
 - b. Correct any errors found using `--dryrun`. Repeat the dry run from the previous step to ensure that no errors were missed.
 - c. When the dry run no longer reports any errors, you can perform the migration using `import cluster`, but without the `--dryrun` option.

3.5.2 Importing a Cluster Into MySQL Cluster Manager: Example

As discussed previously (see [Section 3.5.1, “Importing a Cluster Into MySQL Cluster Manager: Basic Procedure”](#)), importing a standalone or “wild” cluster that was created without the use of MySQL Cluster Manager into the manager requires the completion of four major tasks: create a cluster in MySQL Cluster Manager and update its configuration such that this matches that of the “wild” cluster; prepare the “wild” cluster for MySQL Cluster Manager control; verify all PID files for cluster processes; and performing a dry run and then the actual import using the `import cluster` command. The example provided over the next few sections shows all steps required to perform the importation of a small, standalone MySQL Cluster into MySQL Cluster Manager.

Sample cluster used in example. The “wild” cluster used in this example consists of four nodes—one management node, one SQL node, and two data nodes running `ndbd`. Each of these nodes resides on one of four hosts, all of which are running a recent server release of a typical Linux distribution. The host names for each of these hosts is shown in the following table:

Table 3.4 Nodes used in example cluster

Node type (executable)	Host name
Management node (<code>ndb_mgmd</code>)	<code>alpha</code>
Data node (<code>ndbd</code>)	<code>beta</code>
Data node (<code>ndbd</code>)	<code>gamma</code>
SQL node (<code>mysqld</code>)	<code>delta</code>

We assume that these hosts are on a dedicated network or subnet, and that each of them is running only the MySQL Cluster binaries and applications providing required system and network services. We assume on each host that the MySQL Cluster software has been installed from a release binary archive (see [Installing an NDB Cluster Binary Release on Linux](#)). We also assume that management node is using `/var/lib/mysql-cluster/config.ini` as the cluster's global configuration file, which is shown here:

```
[ndbd default]
DataMemory= 16G
IndexMemory= 12G
NoOfReplicas= 2

[ndb_mgmd]
HostName=alpha
NodeId=50

[ndbd]
NodeId=5
HostName=beta
DataDir=/var/lib/mysql-cluster

[ndbd]
NodeId=6
HostName=gamma
DataDir=/var/lib/mysql-cluster

[mysqld]
NodeId=100
HostName=delta

[ndbapi]
NodeId=101
```

The objective for this example is to bring this cluster, including all of its processes and data, under MySQL Cluster Manager control. This configuration also provides for a “free” SQL node or NDB API application not bound to any particular host; we account for this in the example.

3.5.2.1 Creating and Configuring the Target Cluster

The first task when preparing to import a standalone MySQL Cluster into MySQL Cluster Manager is to create a “target” cluster. Once this is done, we modify the target's configuration until it matches that of the “wild” cluster that we want to import. At a later point in the example, we also show how to test the configuration in a dry run before attempting to perform the actual import.

To create and then configure the target cluster, follow the steps listed here:

- a. Install MySQL Cluster Manager and start `mcmd` on all hosts; we assume that you have installed MySQL Cluster Manager to the recommended location, in this case the directory `/opt/mcm-1.3.6`. (See [Section 2.3, “MySQL Cluster Manager Installation”](#), for more information.) Once you have done this, you can start the `mcm` client (see [Section 3.3, “Starting the MySQL Cluster Manager Client”](#)) on any one of these hosts to perform the next few steps.
- b. Create a MySQL Cluster Manager site encompassing all four of these hosts, using the `create site` command, as shown here:

```
mcm> create site --hosts=alpha,beta,gamma,delta newsite;
+-----+
| Command result |
+-----+
| Site created successfully |
+-----+
1 row in set (0.15 sec)
```

We have named this site `newsite`. You should be able to see it listed in the output of the `list sites` command, similar to what is shown here:

```
mcm> list sites;
+-----+-----+-----+-----+
| Site | Port | Local | Hosts |
+-----+-----+-----+-----+
| newsite | 1862 | Local | alpha,beta,gamma,delta |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

- c. Add a MySQL Cluster Manager package referencing the MySQL Cluster binaries, using the `add package` command; this command's `--basedir` option can be used to point to the correct location. The command shown here creates such a package, named `newpackage`:

```
mcm> add package --basedir=/usr/local/mysql newpackage;
+-----+
| Command result |
+-----+
| Package added successfully |
+-----+
1 row in set (0.70 sec)
```

You do not need to include the `bin` directory containing the MySQL Cluster executables in the `--basedir` path. Since the executables are in `/usr/local/mysql/bin`, it is sufficient to specify `/usr/local/mysql`; MySQL Cluster Manager automatically checks for the binaries in a `bin` directory within the one specified by `--basedir`.

- d. Create the target cluster including at least some of the same processes and hosts used by the standalone cluster. *Do not include any processes or hosts that are not part of this cluster.* In order to prevent potentially disruptive process or cluster operations from interfering by accident with the import process, it is strongly recommended that you create the cluster for import, using the `--import` option for the `create cluster` command.

You must also take care to preserve the correct node ID (as listed in the `config.ini` file shown previously) for each node. In MySQL Cluster Manager 1.3.1 and later, using the `--import` option allows you to specify node IDs under 49 for nodes other than data nodes, which is otherwise prevented when using `create cluster` (the restriction has been lifted since MySQL Cluster Manager 1.3.4).

The following command creates the cluster `newcluster` for import, and includes the management and data nodes, but not the SQL or “free” API node (which we add in the next step):

```
mcm> create cluster --import --package=newpackage \
    --processhosts=ndb_mgmd:50@alpha,ndbd:1@beta,ndbd:2@gamma \
    newcluster;
```

```
+-----+
| Command result |
+-----+
| Cluster created successfully |
+-----+
1 row in set (0.96 sec)
```

You can verify that the cluster was created correctly by checking the output of `show status` with the `--process (-r)` option, like this:

```
mcm> show status -r newcluster;
```

NodeId	Process	Host	Status	Nodegroup	Package
50	ndb_mgmd	alpha	import		newpackage
5	ndbd	beta	import	n/a	newpackage
6	ndbd	gamma	import	n/a	newpackage

```
+-----+
3 rows in set (0.01 sec)
```

- e. If necessary, add any remaining processes and hosts from the “wild” cluster not included in the previous step using one or more `add process` commands. We have not yet accounted for 2 of the nodes from the wild cluster: the SQL node with node ID 100, on host `delta`; and the API node which has node ID 101, and is not bound to any specific host. You can use the following command to add both of these processes to `newcluster`:

```
mcm> add process --processhosts=mysql:100@delta,ndbapi:101@* newcluster;
```

```
+-----+
| Command result |
+-----+
| Process added successfully |
+-----+
1 row in set (0.41 sec)
```

Once again checking the output from `show status -r`, we see that the `mysqld` and `ndbapi` processes were added as expected:

```
mcm> show status -r newcluster;
```

NodeId	Process	Host	Status	Nodegroup	Package
50	ndb_mgmd	alpha	import		newpackage
5	ndbd	beta	import	n/a	newpackage
6	ndbd	gamma	import	n/a	newpackage
100	mysqld	delta	import		newpackage
101	ndbapi	*	import		newpackage

```
+-----+
```

```
5 rows in set (0.08 sec)
```

You can also see that, since `newcluster` was created using the `create cluster` command's `--import` option, the status of all processes in this cluster—including those we just added—is `import`. This means we cannot yet start `newcluster` or any of its processes, as shown here:

```
mcm> start process 50 newcluster;
ERROR 5317 (00MGR): Unable to perform operation on cluster created for import
mcm> start cluster newcluster;
ERROR 5317 (00MGR): Unable to perform operation on cluster created for import
```

The `import` status and its effects on `newcluster` and its cluster processes persist until we have completed importing another cluster into `newcluster`.

Following the execution of the `add process` command shown previously, the target `newcluster` cluster now has the same processes, with the same node IDs, and on the same hosts as the original standalone cluster. We are ready to proceed to the next step.

- f. Duplicate the wild cluster's configuration attributes in the target cluster. In MySQL Cluster Manager 1.3.1 and later, you can handle most of these using the `import config` command. Test out first the effects of the command by running it with the `--dryrun` option (the step only works if you have created the `mcmd` user on the cluster's `mysql` nodes):

```
mcm> import config --dryrun newcluster;
+-----+
| Command result |
+-----+
| Import checks passed. Please check log for settings that will be applied. |
+-----+
1 row in set (5.36 sec)
```



Important

Before executing this command it is necessary to set any non-default ports for `ndb_mgmd` and `mysqld` processes using the `set` command in the `mcm` client.

As indicated by the output from `import config --dryrun`, you can see the configuration attributes and values that would be copied to `newcluster` by the unimpeded command in the agent log file (`mcmd.log`), which by default is created in the MySQL Cluster Manager installation directory. If you open this file in a text editor, you can locate a series of `set` commands that would accomplish this task, similar to what is shown here in *emphasized* texts:

```
2014-03-14 16:05:11.896: (message) [T0x1ad12a0 CMGR ]: Got new message mgr_import_configvalues {84880f7
2014-03-14 16:05:11.896: (message) [T0x1ad12a0 CMGR ]: Got new message mgr_import_configvalues {84880f7
2014-03-14 16:05:11.896: (message) [T0x1ad12a0 CMGR ]: Got new message mgr_import_configvalues {84880f7
2014-03-14 16:05:13.698: (message) [T0x7f4fb80171a0 RECFG]: All utility process have finished
2014-03-14 16:05:13.698: (message) [T0x7f4fb80171a0 RECFG]: Process started : /usr/local/mysql/bin/mysq
2014-03-14 16:05:13.698: (message) [T0x7f4fb80171a0 RECFG]: Spawning mysqld --nodefaults --help --verbo
2014-03-14 16:05:13.904: (message) [T0x7f4fb80171a0 RECFG]: Successfully pulled default configuration f
2014-03-14 16:05:13.905: (warning) [T0x7f4fb80171a0 RECFG]: Failed to remove evsource!
2014-03-14 16:05:15.719: (message) [T0x7f4fb80171a0 RECFG]: All utility process have finished
2014-03-14 16:05:15.725: (message) [T0x7f4fb80171a0 RECFG]: Applying mysqld configuration to cluster...
2014-03-14 16:05:16.186: (message) [T0x1ad12a0 CMGR ]: Got new message mgr_import_configvalues {84880f7
2014-03-14 16:05:16.187: (message) [T0x1ad12a0 CMGR ]: Got new message x_trans {84880f7a 39 0}
2014-03-14 16:05:16.286: (message) [T0x1ad12a0 CMGR ]: Got new message x_trans {84880f7a 40 0}
2014-03-14 16:05:16.286: (message) [T0x7f4fb80171a0 RECFG]: The following will be applied to the curren
set DataDir:ndb_mgmd:50="" newcluster
set IndexMemory:ndbd:5=1073741824 newcluster
set DataMemory:ndbd:5=1073741824 newcluster
set DataDir:ndbd:5=/usr/local/mysql/mysql-cluster/data newcluster
set ThreadConfig:ndbd:5="" newcluster
set IndexMemory:ndbd:6=1073741824 newcluster
```

```

set DataMemory:ndbd:6=1073741824 newcluster
set DataDir:ndbd:6=/usr/local/mysql/mysql-cluster/data newcluster
set ThreadConfig:ndbd:6="" newcluster
set basedir:mysql:100=/usr/local/mysql newcluster
set character_sets_dir:mysql:100=/usr/local/mysql/share/charsets newcluster
set datadir:mysql:100=/usr/local/mysql/data newcluster
set general_log_file:mysql:100=/usr/local/mysql/data/delta.log newcluster
set lc_messages_dir:mysql:100=/usr/local/mysql/share newcluster
set log_error:mysql:100=/usr/local/mysql/data/delta.err newcluster
set ndb_connectstring:mysql:100=alpha newcluster
set ndb_mgmd_host:mysql:100=alpha newcluster
set optimizer_trace:mysql:100=enabled=off,one_line=off newcluster
set pid_file:mysql:100=/usr/local/mysql/data/delta.pid newcluster
set plugin_dir:mysql:100=/usr/local/mysql/lib/plugin newcluster
set report_port:mysql:100=3306 newcluster
set slow_query_log_file:mysql:100=/usr/local/mysql/data/delta-slow.log newcluster
set sql_mode:mysql:100=STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION newcluster

```

Options used at the command line instead of in a configuration file to start a node of the standalone cluster are not imported into the target cluster by the `import config` command; moreover, they will cause one of the following to happen when the `import config --dryrun` is run:

- i. For some options, MySQL Cluster Manager will issue a warning that “Option `<param>` may be removed on next restart of process `<type><nodeid>`,” meaning that those options will not be imported into the target cluster, and thus will not be applied when those nodes are restarted after the import. Here are the lists of such options for each node type:

- For `ndb_mgmd` nodes: `---configdir`, `--initial`, `--log-name`, `--reload`, `--verbose`
- For `ndbd` and `ndbmt` nodes: `--connect-retries`, `--connect-delay`, `--daemon=false`, `--nodaemon`, `--verbose`
- For `mysqld` nodes: `--ndbcluster`, the `--ndbinfo-*` options, `--verbose`, `--datadir`, `--defaults-group-suffix`

When a warning is received for any of these options but you want its value to be included as part of the configuration for your target cluster, you will need to set the option value for your target cluster manually using the `set` command before you perform the final `import cluster` step.

- ii. For some other options, while their values will also not be imported into the target cluster, unlike options described in (i) above, no warnings will be issued for them. Here are lists of such options for each node type:

- For `ndb_mgmd` nodes: `--config-cache`, `--daemon`, `--ndb-nodeid`, `---nodaemon=false`, `--config-file`, `--skip-config-cache`
- For `ndbd` and `ndbmt` nodes: `--daemon`, `--foreground`, `--initial`, `--ndb-connectstring`, `--connect-string`, `--ndb-mgmd-host`, `--ndb-nodeid`, `--nodaemon=false`
- For `mysqld` nodes: `--ndb-connectstring`, `--ndb-mgmd-host`, `--ndb-nodeid`, `--defaults-file`, `--no-defaults`, `--basedir`

- iii. For options that belong to neither of the groups in (i) or (ii) above, having started the standalone cluster's nodes with them at the command line will cause the `import config --dryrun` command to fail with an error, complaining that the options are unsupported. You have to restart

the wild cluster's nodes without those options, and then retry the `import config --dryrun` command.

Assuming that the dry run was successful, you should now be able to import the wild cluster's configuration into `newcluster`, with the command and a result similar to what is shown here:

```
mcm> import config newcluster;
+-----+
| Command result |
+-----+
| Configuration imported successfully. Please manually verify plugin options, abstraction level and def |
+-----+
```

You should check the log from the dry run and the resulting configuration of `newcluster` carefully against the configuration of the wild cluster. If you find any inconsistencies, you must correct these in `newcluster` using the appropriate `set` commands afterwards.

Manual configuration import (MySQL Cluster Manager 1.3.0). In MySQL Cluster Manager 1.3.0, which does not support the `import config` command, it is necessary to copy the wild cluster's configuration manually, using `set` commands in the `mcm` client (once you have obtained the values of any attributes that differ from their defaults). The remainder of this section applies primarily to MySQL Cluster Manager 1.3.0 and the process described here is generally not needed in MySQL Cluster Manager 1.3.1 and later.

MySQL Cluster global configuration data is stored in a file that is usually (but not always) named `config.ini`. Its location on a management node host is arbitrary (there is no default location for it), but if this is not already known, you can determine it by checking—for example, on a typical Linux system—the output of `ps` for the `--config-file` option value that the management node was started with, shown with emphasized text in the output:

```
shell> ps ax | grep ndb_mgmd
18851 ?        Ssl      0:00 ./ndb_mgmd --config-file=/var/lib/mysql-cluster/config.ini
18922 pts/4    S+       0:00 grep --color=auto ndb_mgmd
```

This file uses `INI` format to store global configuration information, and is thus easy to read, or to parse with a script. We start the setup of the target cluster' configuration by checking each section of this file in turn. The first section is repeated here:

```
[ndbd default]
DataMemory= 16G
IndexMemory= 12G
NoOfReplicas= 2
```

The `[ndbd default]` heading indicates that all attributes defined in this section apply to all cluster data nodes. We can set all three attributes listed in this section of the file for all data nodes in `newcluster`, using the `set` command shown here:

```
mcm> set DataMemory:ndbd=16G,IndexMemory:ndbd=12G,NoOfReplicas:ndbd=2 newcluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (0.36 sec)
```

You can verify that the desired changes have taken effect using the `get` command, as shown here:

```
mcm> get DataMemory:ndbd,IndexMemory:ndbd,NoOfReplicas:ndbd newcluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | NodeId1 | Process2 | NodeId2 | Level  | Comment |
+-----+-----+-----+-----+-----+-----+-----+
```

```

| DataMemory | 16G | ndbd | 5 | | | | | Process | | |
| IndexMemory | 12G | ndbd | 5 | | | | | Process | | |
| NoOfReplicas | 2 | ndbd | 5 | | | | | Process | | |
| DataMemory | 16G | ndbd | 6 | | | | | Process | | |
| IndexMemory | 12G | ndbd | 6 | | | | | Process | | |
| NoOfReplicas | 2 | ndbd | 6 | | | | | Process | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.07 sec)

```

The next section in the file is shown here:

```

[ndb_mgmd]
HostName=alpha
NodeId=1

```

This section of the file applies to the management node. We set its `NodeId` and `HostName` attributes previously, when we created `newcluster`. No further changes are required at this time.

The next two sections of the `config.ini` file, shown here, contain configuration values specific to each of the data nodes:

```

[ndbd]
NodeId=5
HostName=beta
DataDir=/var/lib/mysql-cluster

[ndbd]
NodeId=6
HostName=gamma
DataDir=/var/lib/mysql-cluster

```

As was the case for the management node, we already provided the correct node IDs and host names for the data nodes when we created `newcluster`, so only the `DataDir` attribute remains to be set. We can accomplish this by executing the following command in the `mcm` client:

```

mcm> set DataDir:ndbd:5=/var/lib/mysql-cluster,DataDir:ndbd:6=/var/lib/mysql-cluster \
      newcluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (0.42 sec)

```

You may have noticed that we could have set the `DataDir` attribute on the process level using the shorter and simpler command `set DataDir:ndbd=/var/lib/mysql-cluster newcluster`, but since this attribute was defined individually for each data node in the original configuration, we match this scope in the new configuration by setting this attribute for each `ndbd` instance instead. Once again, we check the result using the `mcm` client `get` command, as shown here:

```

mcm> get DataDir:ndbd newcluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name | Value | Process1 | NodeId1 | Process2 | NodeId2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| DataDir | /var/lib/mysql-cluster | ndbd | 5 | | | | |
| DataDir | /var/lib/mysql-cluster | ndbd | 6 | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

```

Configuration attributes for the SQL node are contained the next section of the file, shown here:

```

[mysqld]
NodeId=100

```

```
HostName=delta
```

The `NodeId` and `HostName` attributes were already set when we added the `mysqld` process to `newcluster`, so no additional `set` commands are required at this point. *Keep in mind that there may be additional local configuration values for this `mysqld` that must be accounted for in the configuration we are creating for `newcluster`*; we discuss how to determine these values on the SQL node later in this section.

The remaining section of the file, shown here, contains a section defining attributes for a “free” API node that is not required to connect from any particular host:

```
[ndbapi]
NodeId=101
```

We have already set the `NodeId` and there is no need for a `HostName` for a free process. There are no other attributes that need to be set for this node.

For more information about the MySQL `config.ini` global configuration file, see [NDB Cluster Configuration Files](#), and [NDB Cluster Configuration: Basic Example](#).

As mentioned earlier in this section, each `mysqld` process (SQL node) may have, in addition to any attributes set in `config.ini`, its own configuration data in the form of system variables which are specific to that `mysqld`. These can be set in two ways:

- When the `mysqld` is started, by passing options to it on the command line when invoking it, by reading from an options file (usually named `my.cnf`, or `my.ini` on Windows), or both.
- At runtime using the SQL `SET` statement.

Because the initial values of many options can be changed at runtime, it is recommended that—rather than attempt to read the `my.cnf` or `my.ini` file—you check values for all system variables on each SQL node “live” in the `mysql` client by examining the output of the `SHOW VARIABLES` statement, and execute `set` commands setting each of these values where it differs from the default for that variable on that SQL node.

The `mcm` client can execute a script file containing client commands. The contents of such a script, named `my-commands.mcm`, which contains all commands we executed to create and configure `newcluster`, are shown here:

```
create cluster --import --package=newpackage --processhosts=ndb_mgmd:50@alpha,ndbd:5@beta,ndbd:6@gamma newcluster;
add process --processhosts=mysqld:100@delta,ndbapi:101@* newcluster;
set DataMemory:ndbd=16G,IndexMemory:ndbd=12G,NoOfReplicas:ndbd=2 newcluster;
set DataDir:ndbd:5=/var/lib/mysql-cluster,DataDir:ndbd:6=/var/lib/mysql-cluster newcluster;
```

You can run such a script by invoking the client from the command line with a redirection operator, like this:

```
shell> mcm < my-commands.mcm
```

The name of the script file is completely arbitrary. It must contain valid `mcm` client commands or comments only. (A comment is delimited by a `#` character, and extends from the point in the line where this is found to the end of the line.) Any valid `mcm` client command can be used in such a file. `mcm` must be able to read the file, but the file need not be executable, or readable by any other users.

3.5.2.2 Preparing the Standalone Cluster for Migration

The next step in the import process is to prepare the “wild” cluster for migration. This requires creating an `mcmd` user account with root privileges on all hosts in the cluster; killing any data node angel processes

that may be running; restarting all management nodes without configuration caching; removing cluster processes from control by any system service management facility. More detailed information about performing these tasks is provided in the remainder of this section.



Note

Before proceeding with any migration, the taking of a backup using the `ndb_mgm` client's `START BACKUP` command is strongly recommended.

- a. MySQL Cluster Manager acts through a MySQL user named `mcmd` on each SQL node. It is therefore necessary to create this user and grant root privileges to it. To do this, log in to the SQL node running on host `delta` and execute in the `mysql` client the SQL statements shown here:

```
CREATE USER 'mcmd'@'delta' IDENTIFIED BY 'super';
GRANT ALL PRIVILEGES ON *.* TO 'mcmd'@'delta' WITH GRANT OPTION;
```

Keep in mind that, if the “wild” cluster has more than one SQL node, you must create the `mcmd` user on every one of these nodes.

- b. Kill each data node angel process using the system's facility for doing so. Do not kill any non-angel data node daemons. On a Linux system, you can identify angel processes by matching their process IDs with the owner IDs of the remaining `ndbd` processes in the output of `ps` executed on host `beta` of the example cluster, as shown here, with the relevant process IDs shown in emphasized text:

```
shell> ps -ef | grep ndbd
jon      2023      1  0 18:46 ?          00:00:00 ./ndbd -c alpha
jon      2024 2023    1 18:46 ?          00:00:00 ./ndbd -c alpha
jon      2124 1819    0 18:46 pts/2    00:00:00 grep --color=auto ndbd
```

Use the `kill` command to terminate the process with the indicated process ID, like this:

```
shell> kill -9 2023
```

Verify that the angel process has been killed, and that only one of the two original `ndbd` processes remain, by issuing `ps` again, as shown here:

```
shell> ps -ef | grep ndbd
jon      2024      1  1 18:46 ?          00:00:01 ./ndbd -c alpha
jon      2150 1819    0 18:47 pts/2    00:00:00 grep --color=auto ndbd
```

Now repeat this process from a login shell on host `gamma`, as shown here:

```
shell> ps -ef | grep ndbd
jon      2066      1  0 18:46 ?          00:00:00 ./ndbd -c alpha
jon      2067 2066    1 18:46 ?          00:00:00 ./ndbd -c alpha
jon      3712 1704    0 18:46 pts/2    00:00:00 grep --color=auto ndbd

shell> kill -9 2066

shell> ps -ef | grep ndbd
jon      2067      1  1 18:46 ?          00:00:01 ./ndbd -c alpha
jon      2150 1819    0 18:47 pts/2    00:00:00 grep --color=auto ndbd
```

The wild cluster's data nodes are now ready for migration.

- c. Kill and restart each management node process. When restarting `ndb_mgmd`, its configuration cache must be disabled; since this is enabled by default, you must start the management server with `--config-cache=false`, in addition to any other options that it was previously started with.

**Caution**

Do not use `0` or `OFF` for the value of the `--config-cache` option when restarting `ndb_mgmd` in this step. Using either of these values instead of `false` at this time causes the migration of the management node process to fail at later point in the importation process.

On Linux, we can once again use `ps` to obtain the information we need to accomplish this, this time in a shell on host `alpha`:

```
shell> ps -ef | grep ndb_mgmd
jon  16005      1  1 18:46 ?          00:00:09 ./ndb_mgmd -f /etc/mysql-cluster/config.ini
jon  16401    1819  0 18:58 pts/2      00:00:00 grep --color=auto ndb_mgmd
```

The process ID is 16005, and the management node was started with the `-f` option (the short form for `--config-file`). First, terminate the management using `kill`, as shown here, with the process ID obtained from `ps` previously:

```
shell> kill -9 16005
```

Verify that the management node process was killed, like this:

```
shell> ps -ef | grep ndb_mgmd
jon  16532    1819  0 19:03 pts/2      00:00:00 grep --color=auto ndb_mgmd
```

Now restart the management node as described previously, with the same options that it was started with previously, and with the configuration cache disabled. Change to the directory where `ndb_mgmd` is located, and restart it, like this:

```
shell> ./ndb_mgmd -f /etc/mysql-cluster/config.ini --config-cache=false
MySQL Cluster Management Server mysql-5.6.24-ndb-7.4.6
2013-12-06 19:16:08 [MgmtSrvr] INFO -- Skipping check of config directory since
config cache is disabled.
```

Verify that the process is running as expected, using `ps`:

```
shell> ps -ef | grep ndb_mgmd
jon  17066      1  1 19:16 ?          00:00:01 ./ndb_mgmd -f
/etc/mysql-cluster/config.ini --config-cache=false
jon  17311    1819  0 19:17 pts/2      00:00:00 grep --color=auto ndb_mgmd
```

The management node is now ready for migration.

**Important**

While our example cluster has only a single management node, it is possible for a MySQL Cluster to have more than one. In such cases, you must stop and restart *each* management node process as just described in this step.

- d. Any cluster processes that are under the control of a system boot process management facility, such as `/etc/init.d` on Linux systems or the Services Manager on Windows platforms, should be removed from this facility's control. Consult your system operating documentation for information about how to do this. Be sure not to stop any running cluster processes in the course of doing so.
- e. It is highly recommended that you take a complete backup of the “wild” cluster before proceeding any further, using the `ndb_mgm` client's `START BACKUP` command:

```
ndb_mgm> START BACKUP
Waiting for completed, this may take several minutes
```

```
Node 5: Backup 1 started from node 1
Node 5: Backup 1 started from node 1 completed
StartGCP: 1338 StopGCP: 20134
#Records: 205044 #LogRecords: 10112
Data: 492807474 bytes Log: 317805 bytes
```

It may require some time for the backup to complete, depending on the size of the cluster's data and logs. For `START BACKUP` command options and additional information, see [Using The NDB Cluster Management Client to Create a Backup](#).

3.5.2.3 Verify All Cluster Process PID Files

- a. You must verify that each process in the “wild” cluster has a valid PID file. For purposes of this discussion, a valid PID file has the following characteristics:
 - The filename is `ndb_node_id.pid`, where `node_id` is the node ID used for this process.
 - The file is located in the data directory used by this process.
 - The first line of the file contains the process ID, and only the process ID.

To check the PID file for the management node process, log into the system shell on host `alpha`, and change to the management node's data directory. If this is not specified, the PID file should be created in the same directory that `ndb_mgmd` runs in; change to this directory instead. Then check to see whether the PID file is present using your system's tools for doing this. On Linux, you can use the command shown here:

```
shell> ls ndb_1*
ndb_1_cluster.log  ndb_1_out.log  ndb_1.pid
```

Check the content of the matching `.pid` file using a pager or text editor. We use `more` for this purpose here:

```
shell> more ndb_1.pid
17066
```

The number shown should match the `ndb_mgmd` process ID. We can check this on Linux as before, using `ps`:

```
shell> ps -ef | grep ndb_mgmd
jon  17066  1  1 19:16 ?        00:00:01 ./ndb_mgmd -f /etc/mysql-cluster/config.ini --config-cache=false
jon  17942 1819  0 19:17 pts/2    00:00:00 grep --color=auto ndb_mgmd
```

The management node PID file satisfies the requirements listed at the beginning of this section. Next, we check the PID files for the data nodes, on hosts `beta` and `gamma`. Log into a system shell on `beta`, then obtain the process ID of the `ndbd` process on this host, as shown here:

```
shell> ps -ef | grep ndbd
jon  2024  1  1 18:46 ?        00:00:01 ./ndbd -c alpha
jon  2150 1819  0 18:47 pts/2    00:00:00 grep --color=auto ndbd
```

We observed earlier (see [Section 3.5.2.1, “Creating and Configuring the Target Cluster”](#)) that this node's node ID is 5 and that its `DataDir` is `/var/lib/mysql-cluster`. Check in this directory for the presence of a file named `ndb_5.pid`:

```
shell> ls /var/lib/mysql-cluster/ndb_5.pid
ndb_5.pid
```

Now check the content of this file and make certain that it contains the process ID 2024 on the first line and no other content, like this:

```
shell> more /var/lib/mysql-cluster/ndb_5.pid
2024
```

Similarly, we locate and check the content of the PID file for the remaining data node (node ID 6, data directory `/var/lib/mysql-cluster/`) on host `gamma`:

```
shell> ps -ef | grep ndbd
jon   2067    1  1 18:46 ?        00:00:01 ./ndbd -c alpha
jon   2150  1819  0 18:47 pts/2    00:00:00 grep --color=auto ndbd

shell> ls /var/lib/mysql-cluster/ndb_6.pid
ndb_6.pid

shell> more /var/lib/mysql-cluster/ndb_6.pid
2067
```

The PID file for this data node also meets our requirements, so we are now ready to proceed to the `mysqld` binary running on host `delta`. We handle the PID file for this process in the next step.

- b. If a given process does not have a valid PID file, you must create one for it, or, in some cases, modify the existing one. This is most likely to be a concern when checking PID files for `mysqld` processes, due to the fact that the MySQL Server is customarily started using the startup script `mysqld_safe`, which can start the `mysqld` binary with any number of default options, including the `--pid-file` option. We see that is the case when we check on host `delta` for the running `mysqld` process there (emphasized text):

```
shell> ps -ef | grep mysqld
jon   8782  8520  0 10:30 pts/3    00:00:00 /bin/sh ./mysqld_safe --ndbcluster
--ndb-connectstring=alpha
jon   8893  8782  1 10:30 pts/3    00:00:00 /usr/local/mysql/bin/mysqld
--basedir=/usr/local/mysql --datadir=/usr/local/mysql/data
--plugin-dir=/usr/local/mysql/lib/plugin --ndbcluster --ndb-connectstring=alpha
--log-error=/usr/local/mysql/data/delta.err --pid-file=/usr/local/mysql/data/delta.pid
jon           8947  8520  0 10:30 pts/3    00:00:00 grep --color=auto mysqld

shell> more /usr/local/mysql/data/delta.pid
8893
```

The PID for the SQL node is in an acceptable location (the data directory) and has the correct content, but has the wrong name.

You can create a correct PID file in either of two locations—in the process data directory, or in the directory `mcm_dir/clusters/cluster_name/pid/` on the same host as the process, where `mcm_dir` is the MySQL Cluster Manager installation directory, and `cluster_name` is the name of the cluster. In this case, since the existing PID file is otherwise correct, it is probably easiest just to copy it to a correctly named file in the same directory incorporating the node ID (100), like this:

```
shell> cp /usr/local/mysql/data/delta.pid /usr/local/mysql/data/ndb_100.pid
```

Another alternative is to create and write a completely new PID file to the proper location in the MySQL Cluster Manager installation directory, as shown here:

```
shell> echo '8893' > /opt/mcm-1.3.6/clusters/newcluster/pid/ndb_100.pid

shell> more /opt/mcm-1.3.6/clusters/newcluster/pid/ndb_100.pid
8893
```

`ndbapi` processes running under MySQL Cluster Manager do not require PID files, so we have completed this step of the import, and we should be ready for a test or “dry run” of the migration. We perform this test in the next step.

3.5.2.4 Testing and Migrating the Standalone Cluster

Testing and performing and performing the migration of a standalone MySQL Cluster into MySQL Cluster Manager consists of the following steps:

1. Perform a test run of the proposed import using `import cluster` with the `--dryrun` option. When this option is used, MySQL Cluster Manager checks for mismatched configuration attributes, missing or invalid processes or hosts, missing or invalid PID files, and other errors, and warns of any it finds, but does not actually perform any migration of processes or data.

```
mcm> import cluster --dryrun newcluster;
ERROR 5302 (00MGR): No access for user mcmd to mysqld 100 in cluster newcluster.
Please verify user access and grants adhere to documented requirements.
```

We omitted a crucial step earlier: we apparently neglected to create the `mcmd` superuser account needed on all SQL nodes in the “wild” cluster to bring them under control of MySQL Cluster Manager. In this case, there is only one SQL node, running on `delta`. Log into this SQL node as the MySQL `root` user, and create the `mcmd` account in the `mysql` client, as shown here:

```
shell> ./mysql -uroot -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.24-ndb-7.4.6 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE USER 'mcmd'@'localhost' IDENTIFIED BY 'super';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.*
-> TO 'mcmd'@'localhost' IDENTIFIED BY 'super'
-> WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'mcmd'@'localhost'\G
***** 1. row *****
Grants for mcmd@localhost: GRANT ALL PRIVILEGES ON *.* TO 'mcmd'@'localhost'
IDENTIFIED BY PASSWORD '*F85A86E6F55A370C1A115F696A9AD71A7869DB81' WITH GRANT OPTION
1 row in set (0.00 sec)
```

Having corrected this issue on the SQL node, we repeat the dry run in the `mcm` client:

```
mcm> import cluster --dryrun newcluster;
ERROR 5310 (00MGR): Process ndb_mgmd 50 reported 6 processes, while 5 processes
are configured for cluster newcluster
```

This error means that there are one or more cluster processes not accounted for the configuration for the target cluster. Checking the contents of the file `/etc/mysql-cluster/config.ini` on host `alpha`, we see that we overlooked a section in it earlier. This section is shown here:

```
[mysqld]
NodeId=102
```

To address this discrepancy, we need to add another “free” `ndbapi` process to `newcluster`, which we can do by executing the following `add process` command in the `mcm` client:

```
mcm> add process -R ndbapi:102@* newcluster;
+-----+
| Command result |
+-----+
| Process added successfully |
+-----+
1 row in set (0.38 sec)
```

You can verify this by checking the output of `show status -r` command, as shown here:

```
mcm> show status -r newcluster;
+-----+-----+-----+-----+-----+-----+
| NodeId | Process | Host | Status | Nodegroup | Package |
+-----+-----+-----+-----+-----+-----+
| 50      | ndb_mgmd | alpha | import |            | newpackage |
| 5       | ndbd     | beta  | import | n/a       | newpackage |
| 6       | ndbd     | gamma | import | n/a       | newpackage |
| 100     | mysqld   | delta | import |            | newpackage |
| 101     | ndbapi   | *     | import |            |            |
| 102     | ndbapi   | *     | import |            |            |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.11 sec)
```

Now we can run another test, using `import cluster` with the `--dryrun` as we did previously:

```
mcm> import cluster --dryrun newcluster;
```

- Continue to correct any errors or other discrepancies found using `--dryrun`, repeating the dry run shown in the previous step to ensure that no errors were missed. The following list contains some common errors you may encounter, and their likely causes:
 - MySQL Cluster Manager requires a specific MySQL user and privileges to manage SQL nodes. If the `mcmd` MySQL user account is not set up properly, you may see `No access for user...`, `Incorrect grants for user...`, or possibly other errors. See [Section 3.5.2.2, "Preparing the Standalone Cluster for Migration"](#).
 - As described previously, each cluster process (other than a process whose type is `ndbapi`) being brought under MySQL Cluster Manager control must have a valid PID file. Missing, misnamed, or invalid PID files can produce errors such as `PID file does not exist for process...`, `PID ... is not running ...`, and `PID ... is type ...`. See [Section 3.5.2.3, "Verify All Cluster Process PID Files"](#).
 - Process version mismatches can also produce seemingly random errors whose cause can sometime prove difficult to track down. Ensure that all nodes are supplied with the correct release of the MySQL Cluster software, and that it is the same release and version of the software.
 - Each data node angel process in the standalone cluster must be stopped prior to import. A running angel process can cause errors such as `Angel process pid exists ...` or `Process pid is an angel process for ...`. See [Section 3.5.2.2, "Preparing the Standalone Cluster for Migration"](#).
 - The number of processes, their types, and the hosts where they reside in the standalone cluster must be reflected accurately when creating the target site, package, and cluster for import. Otherwise, errors such as `Process id reported # processes ...`, `Process id ... does not match configured process ...`, `Process idnot configured ...`, and `Process iddoes not match configured process ...`. See [Section 3.5.2.1, "Creating and Configuring the Target Cluster"](#).

- Other factors that can cause specific errors include processes in the wrong state, processes that were started with unsupported command-line options (see [Section 3.5.2.1, “Creating and Configuring the Target Cluster”](#) for details) or without required options, and processes having the wrong process ID, or using the wrong node ID.
3. When `import cluster --dryrun` no longer warns of any errors, you can perform the import with the `import cluster` command, this time omitting the `--dryrun` option.

3.6 MySQL Cluster Backup and Restore Using MySQL Cluster Manager

This section describes usage of the [NDB](#) native backup and restore functionality implemented in MySQL Cluster Manager, to perform a number of common tasks.

3.6.1 Requirements for Backup and Restore

This section provides information about basic requirements for performing backup and restore operations using MySQL Cluster Manager.

Requirements for MySQL Cluster backup. Basic requirements for performing MySQL backups using MySQL Cluster Manager are minimal. At least one data node in each node group must be running, and there must be sufficient disk space on the node file systems. Partial backups are not supported.

Requirements for MySQL Cluster restore. Restoring a MySQL Cluster using MySQL Cluster Manager is subject to the following conditions:

- A complete restore requires that all data nodes are up and running, and that all files belonging to a given backup are available.
- A partial restore is possible, but must be specified as such. This can be accomplished using the `restore cluster` client command with its `--skip-nodeid` option.
- In the event that data nodes have been added to the cluster since the backup was taken, only those data nodes for which backup files exist are restored. In such cases data is not automatically distributed to the new nodes, and, following the restore, you must redistribute the data manually by issuing an `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement in the `mysql` client for each [NDB](#) table in the cluster. See [Adding NDB Cluster Data Nodes Online: Basic procedure](#), for more information.

3.6.2 Basic MySQL Cluster Backup and Restore Using MySQL Cluster Manager

This section describes backing up and restoring a MySQL Cluster, with examples of complete and partial restore operations. Note that the `backup cluster` and `restore cluster` commands work with [NDB](#) tables only; tables using other MySQL storage engines (such as [InnoDB](#) or [MyISAM](#)) are ignored.

For purposes of example, we use a MySQL Cluster named `mycluster` whose processes and status can be seen here:

```
mcm> show status -r mycluster;
+-----+-----+-----+-----+-----+-----+
| NodeId | Process | Host      | Status | Nodegroup | Package |
+-----+-----+-----+-----+-----+-----+
| 49     | ndb_mgmd | tonfisk   | running |           | mypackage |
| 1      | ndbd     | tonfisk   | running | 0         | mypackage |
| 2      | ndbd     | tonfisk   | running | 0         | mypackage |
```

```

| 50 | mysqld | tonfisk | running | | mypackage |
| 51 | mysqld | tonfisk | running | | mypackage |
| 52 | ndbapi | *tonfisk | added | | |
| 53 | ndbapi | *tonfisk | added | | |
+-----+-----+-----+-----+-----+
7 rows in set (0.08 sec)

```

You can see whether there are any existing backups of `mycluster` using the `list backups` command, as shown here:

```

mcm> list backups mycluster;
+-----+-----+-----+-----+-----+
| BackupId | NodeId | Host | Timestamp | Comment |
+-----+-----+-----+-----+-----+
| 1 | 1 | tonfisk | 2012-12-04 12:03:52 | |
| 1 | 2 | tonfisk | 2012-12-04 12:03:52 | |
| 2 | 1 | tonfisk | 2012-12-04 12:04:15 | |
| 2 | 2 | tonfisk | 2012-12-04 12:04:15 | |
| 3 | 1 | tonfisk | 2012-12-04 12:17:41 | |
| 3 | 2 | tonfisk | 2012-12-04 12:17:41 | |
+-----+-----+-----+-----+-----+
6 rows in set (0.12 sec)

```

Simple backup. To create a backup, use the `backup cluster` command with the name of the cluster as an argument, similar to what is shown here:

```

mcm> backup cluster mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Backup completed successfully |
+-----+-----+
1 row in set (3.31 sec)

```

`backup cluster` requires only the name of the cluster to be backed up as an argument; for information about additional options supported by this command, see [Section 4.7.2, “The backup cluster Command”](#). To verify that a new backup of `mycluster` was created with a unique ID, check the output of `list backups`, as shown here (where the rows corresponding to the new backup files are indicated with emphasized text):

```

mcm> list backups mycluster;
+-----+-----+-----+-----+-----+
| BackupId | NodeId | Host | Timestamp | Comment |
+-----+-----+-----+-----+-----+
| 1 | 1 | tonfisk | 2012-12-04 12:03:52 | |
| 1 | 2 | tonfisk | 2012-12-04 12:03:52 | |
| 2 | 1 | tonfisk | 2012-12-04 12:04:15 | |
| 2 | 2 | tonfisk | 2012-12-04 12:04:15 | |
| 3 | 1 | tonfisk | 2012-12-04 12:17:41 | |
| 3 | 2 | tonfisk | 2012-12-04 12:17:41 | |
| / 4 | / 1 | / tonfisk | / 2012-12-12 14:24:35 | / |
| / 4 | / 2 | / tonfisk | / 2012-12-12 14:24:35 | / |
+-----+-----+-----+-----+-----+
8 rows in set (0.04 sec)

```

If you attempt to create a backup of a MySQL Cluster in which each node group does not have at least one data node running, `backup cluster` fails with the error `Backup cannot be performed as processes are stopped in cluster cluster_name`.

Simple complete restore. To perform a complete restore of a MySQL Cluster from a backup with a given ID, follow the steps listed here:

1. Identify the backup to be used.

In this example, we use the backup having the ID 4, that was created for `mycluster` previously in this section.

2. Wipe the MySQL Cluster data.

The simplest way to do this is to stop and then perform an initial start of the cluster as shown here, using `mycluster`:

```
mcm> stop cluster mycluster;
+-----+
| Command result |
+-----+
| Cluster stopped successfully |
+-----+
1 row in set (15.24 sec)

mcm> start cluster --initial mycluster;
+-----+
| Command result |
+-----+
| Cluster started successfully |
+-----+
1 row in set (34.47 sec)
```

3. Restore the backup.

This is done using the `restore cluster` command, which requires the backup ID and the name of the cluster as arguments. Thus, you can restore backup 4 to `mycluster` as shown here:

```
mcm> restore cluster --backupid=4 mycluster;
+-----+
| Command result |
+-----+
| Restore completed successfully |
+-----+
1 row in set (16.78 sec)
```

Partial restore—missing images. It is possible using MySQL Cluster Manager to perform a partial restore of a MySQL Cluster—that is, to restore from a backup in which backup images from one or more data nodes are not available. This is required if we wish to restore `mycluster` to backup number 6, since an image for this backup is available only for node 1, as can be seen in the output of `list backups` in the `mcm` client (emphasized text):

```
mcm> list backups mycluster;
+-----+-----+-----+-----+-----+
| BackupId | NodeId | Host      | Timestamp          | Comment |
+-----+-----+-----+-----+-----+
| 1         | 1       | tonfisk   | 2012-12-04 12:03:52 |         |
| 1         | 2       | tonfisk   | 2012-12-04 12:03:52 |         |
| 2         | 1       | tonfisk   | 2012-12-04 12:04:15 |         |
| 2         | 2       | tonfisk   | 2012-12-04 12:04:15 |         |
| 3         | 1       | tonfisk   | 2012-12-04 12:17:41 |         |
| 3         | 2       | tonfisk   | 2012-12-04 12:17:41 |         |
| 4         | 1       | tonfisk   | 2012-12-12 14:24:35 |         |
| 4         | 2       | tonfisk   | 2012-12-12 14:24:35 |         |
| 5         | 1       | tonfisk   | 2012-12-12 14:31:31 |         |
| 5         | 2       | tonfisk   | 2012-12-12 14:31:31 |         |
| 6         | 1       | tonfisk   | 2012-12-12 14:32:09 |         |
+-----+-----+-----+-----+-----+
11 rows in set (0.08 sec)
```

To perform a restore of only those nodes for which we have images (in this case, node 1 only), we can use the `--skip-nodeid` option when executing a `restore cluster` command. This option causes one or

more nodes to be skipped when performing the restore. Assuming that `mycluster` has been cleared of data (as described earlier in this section), we can perform a restore that skips node 2 as shown here:

```
mcm> restore cluster --backupid=6 --skip-nodeid=2 mycluster;
+-----+
| Command result |
+-----+
| Restore completed successfully |
+-----+
1 row in set (17.06 sec)
```

Because we excluded node 2 from the restore process, no data has been distributed to it. To cause MySQL Cluster data to be distributed to any such excluded or skipped nodes following a partial restore, it is necessary to redistribute the data manually by executing an `ALTER ONLINE TABLE ... REORGANIZE PARTITION` statement in the `mysql` client for each NDB table in the cluster. To obtain a list of NDB tables from the `mysql` client, you can use multiple `SHOW TABLES` statements or a query such as this one:

```
SELECT CONCAT('`', TABLE_SCHEMA, '`', TABLE_NAME)
FROM INFORMATION_SCHEMA.TABLES
WHERE ENGINE='ndbcluster';
```

You can generate the necessary SQL statements using a more elaborate version of the query just shown, such the one employed here:

```
mysql> SELECT
->   CONCAT('ALTER ONLINE TABLE `', TABLE_SCHEMA,
->         '`', TABLE_NAME, '` REORGANIZE PARTITION;')
->   AS Statement
-> FROM INFORMATION_SCHEMA.TABLES
-> WHERE ENGINE='ndbcluster';
+-----+
| Statement |
+-----+
| ALTER ONLINE TABLE `mysql`.`ndb_apply_status` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `mysql`.`ndb_index_stat_head` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `mysql`.`ndb_index_stat_sample` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `db1`.`n1` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `db1`.`n2` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `db1`.`n3` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `test`.`n1` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `test`.`n2` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `test`.`n3` REORGANIZE PARTITION; |
| ALTER ONLINE TABLE `test`.`n4` REORGANIZE PARTITION; |
+-----+
10 rows in set (0.09 sec)
```

Partial restore—data nodes added. A partial restore can also be performed when new data nodes have been added to a MySQL Cluster following a backup. In this case, you can exclude the new nodes using `--skip-nodeid` when executing the `restore cluster` command. Consider the MySQL Cluster named `mycluster` as shown in the output of the following `show status` command:

```
mcm> show status -r mycluster;
+-----+-----+-----+-----+-----+-----+
| NodeId | Process | Host | Status | Nodegroup | Package |
+-----+-----+-----+-----+-----+-----+
| 49 | ndb_mgmd | tonfisk | stopped | | mypackage |
| 1 | ndbd | tonfisk | stopped | 0 | mypackage |
| 2 | ndbd | tonfisk | stopped | 0 | mypackage |
| 50 | mysqld | tonfisk | stopped | | mypackage |
| 51 | mysqld | tonfisk | stopped | | mypackage |
| 52 | ndbapi | *tonfisk | added | | |
| 53 | ndbapi | *tonfisk | added | | |
+-----+-----+-----+-----+-----+-----+
```

```
7 rows in set (0.03 sec)
```

The output of `list backups` shows us the available backup images for this cluster:

```
mcm> list backups mycluster;
+-----+-----+-----+-----+-----+
| BackupId | NodeId | Host      | Timestamp           | Comment |
+-----+-----+-----+-----+-----+
| 1         | 1       | tonfisk   | 2012-12-04 12:03:52 |         |
| 1         | 2       | tonfisk   | 2012-12-04 12:03:52 |         |
| 2         | 1       | tonfisk   | 2012-12-04 12:04:15 |         |
| 2         | 2       | tonfisk   | 2012-12-04 12:04:15 |         |
| 3         | 1       | tonfisk   | 2012-12-04 12:17:41 |         |
| 3         | 2       | tonfisk   | 2012-12-04 12:17:41 |         |
| 4         | 1       | tonfisk   | 2012-12-12 14:24:35 |         |
| 4         | 2       | tonfisk   | 2012-12-12 14:24:35 |         |
+-----+-----+-----+-----+-----+
```

```
8 rows in set (0.06 sec)
```

Now suppose that, at a later point in time, 2 data nodes have been added to `mycluster` using an `add process` command. The `show status` output for `mycluster` now looks like this:

```
mcm> show status -r mycluster;
+-----+-----+-----+-----+-----+-----+
| NodeId | Process | Host      | Status | Nodegroup | Package |
+-----+-----+-----+-----+-----+-----+
| 49     | ndb_mgmd | tonfisk   | running |           | mypackage |
| 1      | ndbd     | tonfisk   | running | 0         | mypackage |
| 2      | ndbd     | tonfisk   | running | 0         | mypackage |
| 50     | mysqld   | tonfisk   | running |           | mypackage |
| 51     | mysqld   | tonfisk   | running |           | mypackage |
| 52     | ndbapi   | *tonfisk  | added   |           |           |
| 53     | ndbapi   | *tonfisk  | added   |           |           |
| 3      | ndbd     | tonfisk   | running | 1         | mypackage |
| 4      | ndbd     | tonfisk   | running | 1         | mypackage |
+-----+-----+-----+-----+-----+-----+
```

```
9 rows in set (0.01 sec)
```

Since nodes 3 and 4 were not included in the backup, we need to exclude them when performing the restore. You can cause `restore cluster` to skip multiple data nodes by specifying a comma-separated list of node IDs with the `--skip-nodeid` option. Assume that we have just cleared `mycluster` of MySQL Cluster data using the `mcm` client commands `stop cluster` and `start cluster --initial` as described previously in this section; then we can restore `mycluster` (now having 4 data nodes numbered 1, 2, 3, and 4) from backup number 4 (made when `mycluster` had only 2 data nodes numbered 1 and 2) as shown here:

```
mcm> restore cluster --backupid=4 --skip-nodeid=3,4 mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Restore completed successfully |
+-----+-----+
1 row in set (17.61 sec)
```

No data is distributed to the skipped (new) nodes; you must force nodes 3 and 4 to be included in a redistribution of the data using `ALTER ONLINE TABLE ... REORGANIZE PARTITION` as described previously in this section.

3.7 Backing Up and Restoring MySQL Cluster Manager Agents

This section explains how to back up configuration data for `mcmd` agents and how to restore the backed-up agent data. Used together with the `backup cluster` command, the `backup agents` command allows you to backup and restore a complete cluster-plus-manager setup.

If no host names are given with the `backup agents` command, backups are created for all agents of the site:

```
mcm> backup agents mysite;
+-----+
| Command result |
+-----+
| Agent backup created successfully |
+-----+
1 row in set (0.07 sec)
```

To backup one or more specific agents, specify them with the `--hosts` option:

```
mcm> backup agents --hosts=tonfisk mysite;
+-----+
| Command result |
+-----+
| Agent backup created successfully |
+-----+
1 row in set (0.07 sec)
```

If no site name is given, only the agent that the `mcm` client is connected to is backed up.

The backup for each agent includes the following contents from the agent repository (`mcm_data` folder):

- The `rep` subfolder
- The metadata files `high_water_mark` and `repchksum`

The repository is locked while the backup are in progress, to avoid creating an inconsistent backup. The backup for each agent is created in a subfolder named `rep_backup/timestamp` under the agent's `mcm_data` folder, with `timestamp` reflecting the time the backup began. If you want the backup to be at another place, create a soft link from `mcm_data/rep_backup` to your desired storage location.

To restore the backup for an agent:

- Wipe the contents of the agent's `mcm_data/rep` folder
- Delete the metadata files `high_water_mark` and `repchksum` from the `mcm_data` folder
- Copy the contents in the `mcm_data/rep_backup/timestamp/rep` folder back into the `mcm_data/rep` folder
- Copy the metadata files `high_water_mark` and `repchksum` from the `mcm_data/rep_backup/timestamp` folder back into the `mcm_data` folder
- Restart the agent

The steps are illustrated below:

```
mysql@tonfisk$ cd mcm_data
mysql@tonfisk$ cp mcm_data/rep_backup/timestamp/rep/* ./rep/
mysql@tonfisk$ cp mcm_data/rep_backup/timestamp/high_water_mark ./
mysql@tonfisk$ cp mcm_data/rep_backup/timestamp/repchksum ./
mysql@tonfisk$ mcm1.3.6/bin/mcmd
```

The backup may be manually restored on just one, or more than one agents. If backup is restored for only one agent on, say, host A, host A will contact the other agents of the site to make them recover

their repositories from host A using the usual mechanism for agent recovery. If all agents on all hosts are restored and restarted manually, the situation will be similar to the normal restarting all agents after stopping them at slightly different points in time.

If configuration changes has been made to the cluster since the restored backup was created, the same changes must be made again after the agent restores have been completed, to ensure that the agents' configurations match those of the actual running cluster. For example: sometime after a backup was done, a `set MaxNoOfTables:ndbmttd=500 mycluster` command was issued and soon afterward, something happened and corrupted the agent repository; after the agent backup was restored, the same `set` command has to be rerun in order to update the `mcmd` agents' configurations. While the command does not effectively change anything on the cluster itself, after it has been run, a rolling restart of the cluster processes using the `restart cluster` command is still required.

3.8 Setting Up MySQL Cluster Replication with MySQL Cluster Manager

This section provides sample steps for setting up a MySQL Cluster replication with a single replication channel using the MySQL Cluster Manager.

Before trying the following steps, it is recommended that you first read [NDB Cluster Replication](#) to familiarize yourself with the concepts, requirements, operations, and limitations of MySQL Cluster replication.

1. Create and start a source cluster:

```
mcm> create site --hosts=tonfisk msite;

mcm> add package --basedir=/usr/local/cluster-mgt/cluster-7.3.2 7.3.2;

mcm> create cluster -P 7.3.2 -R \
    ndb_mgmd@tonfisk,ndbmttd@tonfisk,ndbmttd@tonfisk,mysqld@tonfisk,mysqld@tonfisk,ndbapi@*,ndbapi@* \
    source;

mcm> set portnumber:ndb_mgmd=4000 source;

mcm> set port:mysql:51=3307 source;

mcm> set port:mysql:50=3306 source;

mcm> set server_id:mysql:50=100 source;

mcm> set log_bin:mysql:50=binlog source;

mcm> set binlog_format:mysql:50=ROW source;

mcm> set ndb_connectstring:mysql:50=tonfisk:4000 source;

mcm> start cluster source;
```

2. Create and start a replica cluster (we begin with creating a new site called “ssite” just for the replica cluster; you can also skip that and put the source and replica cluster hosts under the same site instead):

```
mcm> create site --hosts=flundra ssite;

mcm> add package --basedir=/usr/local/cluster-mgt/cluster-7.3.2 7.3.2;

mcm> create cluster -P 7.3.2 -R \
    ndb_mgmd@flundra,ndbmttd@flundra,ndbmttd@flundra,mysqld@flundra,mysqld@flundra,ndbapi@*,ndbapi@* \
    replica;
```

```
mcm> set portnumber:ndb_mgmd=4000 replica;
mcm> set port:mysqld:50=3306 replica;
mcm> set port:mysqld:51=3307 replica;
mcm> set server_id:mysqld:50=101 replica;
mcm> set ndb_connectstring:mysqld:50=flundra:4000 replica;
mcm> set slave_skip_errors:mysqld=all replica;
mcm> start cluster replica;
```

3. Create a replica account (with the user name “myreplica” and password “mypw”) on the source cluster with the appropriate privilege by logging into the source replication client (`mysqlM`) and issuing the following statements:

```
mysqlM> GRANT REPLICATION SLAVE ON *.* TO 'myreplica'@'flundra'
-> IDENTIFIED BY 'mypw';
```

4. Log into the replica cluster client (`mysqlS`) and issue the following statements:

```
mysqlS> CHANGE MASTER TO
-> MASTER_HOST='tonfisk',
-> MASTER_PORT=3306,
-> MASTER_USER='myreplica',
-> MASTER_PASSWORD='mypw';
```

5. Start replication by issuing the following statement with the replica cluster client:

```
mysqlS> START SLAVE;
```

The above example assumes that the source and replica clusters are created at about the same time, with no data on both before replication starts. If the source cluster has already been operating and has data on it when the salve cluster is created, after step 3 above, follow these steps to transfer the data from the source cluster to the replica cluster and prepare the replica cluster for replication:

1. Back up your source cluster using the `backup cluster` command of MySQL Cluster Manager:

```
mcm> backup cluster source;
```



Note

Only `NDB` tables are backed up by the command; tables using other MySQL storage engines are ignored.

2. Look up the backup ID of the backup you just made by listing all backups for the source cluster:

```
mcm> list backups source;
+-----+-----+-----+-----+-----+
| BackupId | NodeId | Host      | Timestamp           | Comment |
+-----+-----+-----+-----+-----+
| 1         | 1      | tonfisk   | 2014-10-17 20:03:23 |         |
| 1         | 2      | tonfisk   | 2014-10-17 20:03:23 |         |
| 2         | 1      | tonfisk   | 2014-10-17 20:09:00 |         |
| 2         | 2      | tonfisk   | 2014-10-17 20:09:00 |         |
+-----+-----+-----+-----+-----+
```

From the output, you can see that the latest backup you created has the backup ID “2”, and backup data exists for node “1” and “2”.

- Using the backup ID and the related node IDs, identify the backup files just created under `/mcm_data/clusters/cluster_name/node_id/data/BACKUP/BACKUP-backup_id/` in the source cluster's installation directory (in this case, the files under the `/mcm_data/clusters/source/1/data/BACKUP/BACKUP-2` and `/mcm_data/clusters/source/2/data/BACKUP/BACKUP-2`), and copy them over to the equivalent places for the replica cluster (in this case, `/mcm_data/clusters/replica/1/data/BACKUP/BACKUP-2` and `/mcm_data/clusters/replica/2/data/BACKUP/BACKUP-2` under the replica cluster's installation directory). After the copying is finished, use the following command to check that the backup is now available for the replica cluster:

```
mcm> list backups replica;
```

BackupId	NodeId	Host	Timestamp	Comment
2	1	flundra	2014-10-17 21:19:00	
2	2	flundra	2014-10-17 21:19:00	

- Restore the backed up data to the replica cluster (note that you need an unused `ndbapi` slot for the `restore cluster` command to work):

```
mcm> restore cluster --backupid=2 replica;
```

- On the source cluster client, use the following command to identify the correct binary log file and position for replication to start:

```
mysqlM> SHOW MASTER STATUS\G;
***** 1. row *****
      File: binlog.000017
      Position: 2857
      Binlog_Do_DB:
      Binlog_Ignore_DB:
      Executed_Gtid_Set:
```

- On the replica cluster client, provide to the replica cluster the information of the source cluster, including the binary log file name (with the `MASTER_LOG_FILE` option) and position (with the `MASTER_LOG_POS` option) you just discovered in step 5 above:

```
mysqlS> CHANGE MASTER TO
-> MASTER_HOST='tonfisk',
-> MASTER_PORT=3306,
-> MASTER_USER='myreplica',
-> MASTER_PASSWORD='mypw',
-> MASTER_LOG_FILE='binlog.000017',
-> MASTER_LOG_POS=2857;
```

- Start replication by issuing the following statement with the replica cluster client:

```
mysqlS> START SLAVE;
```

As an alternative to these steps, you can also follow the steps described in [NDB Cluster Backups With NDB Cluster Replication](#) to copy the data from the source to the replica and to specify the binary log file and position for replication to start.

Chapter 4 MySQL Cluster Manager Client Commands

Table of Contents

4.1 Online Help and Information Commands	71
4.2 MySQL Cluster Manager Site and Agent Commands	78
4.2.1 The <code>add hosts</code> Command	78
4.2.2 The <code>remove hosts</code> Command	79
4.2.3 The <code>change log-level</code> Command	79
4.2.4 The <code>rotate log</code> Command	80
4.2.5 The <code>collect logs</code> Command	81
4.2.6 The <code>create site</code> Command	81
4.2.7 The <code>delete site</code> Command	82
4.2.8 The <code>list sites</code> Command	83
4.2.9 The <code>list hosts</code> Command	83
4.2.10 The <code>show settings</code> Command	84
4.2.11 The <code>stop agents</code> Command	84
4.2.12 The <code>version</code> Command	85
4.2.13 The <code>show warnings</code> Command	85
4.3 MySQL Cluster Manager Package Commands	86
4.3.1 The <code>add package</code> Command	86
4.3.2 The <code>delete package</code> Command	88
4.3.3 The <code>list packages</code> Command	89
4.4 MySQL Cluster Manager Cluster Commands	90
4.4.1 The <code>create cluster</code> Command	90
4.4.2 The <code>delete cluster</code> Command	94
4.4.3 The <code>list clusters</code> Command	95
4.4.4 The <code>list nextnodeids</code> Command	95
4.4.5 The <code>restart cluster</code> Command	95
4.4.6 The <code>show status</code> Command	96
4.4.7 The <code>start cluster</code> Command	100
4.4.8 The <code>stop cluster</code> Command	101
4.4.9 The <code>upgrade cluster</code> Command	102
4.5 MySQL Cluster Manager Configuration Commands	103
4.5.1 The <code>get</code> Command	106
4.5.2 The <code>reset</code> Command	115
4.5.3 The <code>set</code> Command	122
4.6 MySQL Cluster Manager Process Commands	128
4.6.1 The <code>add process</code> Command	128
4.6.2 The <code>change process</code> Command	131
4.6.3 The <code>list processes</code> Command	134
4.6.4 The <code>start process</code> Command	134
4.6.5 The <code>stop process</code> Command	135
4.6.6 The <code>remove process</code> Command	135
4.7 MySQL Cluster Manager Backup and Restore Commands	137
4.7.1 The <code>abort backup</code> Command	137
4.7.2 The <code>backup cluster</code> Command	137
4.7.3 The <code>list backups</code> Command	138
4.7.4 The <code>restore cluster</code> Command	138
4.7.5 The <code>backup agents</code> Command	140
4.8 MySQL Cluster Manager Cluster Importation Commands	141
4.8.1 The <code>import cluster</code> Command	141

Identifiers in client commands

Case-sensitivity rules for client commands

Options common to client commands

The sections in this chapter describe commands used in the MySQL Cluster Manager 1.3.6 client for tasks such as defining sites, packages, and MySQL Cluster instances (“clusters”); configuring a MySQL Cluster; and getting the status of a running MySQL Cluster. These commands are issued to the management agent using the `mysql` client program included with the MySQL Cluster distribution (for information about the `mysql` client not specific to using MySQL Cluster Manager, see [mysql — The MySQL Command-Line Client](#)). Each MySQL Cluster Manager client command takes the form shown here:

```
instruction [options] [arguments]

options:
  option [option] [...]

option:
  --option-long-name[=value-list]
  | -option-short-name [value-list]

value-list:
  value[,value[,...]]

arguments:
  argument [argument] [...]
```

Consider the following MySQL Cluster Manager command, which starts a MySQL Cluster named `mycluster` and backgrounds the deletion process so that the client can be used to execute other commands in the meantime, without having to wait on the `start cluster` command to complete:

```
start cluster --background mycluster;
```

In this example, the command contains a `start cluster` instruction. An instruction consists of one or two keywords, such as `set`, or `show status`. This instruction is modified by the `--background` option which follows it; however, this option assigns no values.

Most command options have short forms, consisting of single letters, in addition to their long forms. Using the short form of the `--background` option, the previous example could also be written like this:

```
start cluster -B mycluster;
```

The long form of an option must be preceded by a double dash (`--`), and is not case-sensitive (lower case being the canonical form). The short form of an option must be preceded by a single dash (`-`), and is case-sensitive. In either case, the dash character or characters must come immediately before the option name, and there must be no space characters between them. Otherwise, the MySQL Cluster Manager client cannot parse the command correctly. More information about long and short forms of options is given later in this section.



Important

Do not confuse options given to MySQL Cluster Manager client commands with `mysql` client options. A MySQL Cluster Manager client command option is always employed as part of a MySQL Cluster Manager client command; it is *not* passed to the `mysql` client when invoking it.

In addition, you cannot issue queries or other SQL statements in the MySQL Cluster Manager client. These are not recognized by the client, and are rejected with an error. The converse of this is also true: MySQL Cluster Manager client commands are not recognized by the standard `mysql` client.

The instruction just shown takes the argument `mysite`. The argument is usually an identifier that names the object to be effected; in this case, the command deletes the site whose name matches the argument. (For more information, see [Section 4.2.6, “The `create site` Command”](#).)

Changes in client commands in version 1.3.4. An additional `--verbose` option has been added in version 1.3.4 to the `create cluster` and `add process` commands. In both cases, using the option causes the command to return a list of the MySQL Cluster processes affected by the command; this includes their node IDs, process types, and the hosts where they are located.

Identifiers in client commands.

A legal MySQL Cluster Manager identifier consists of any sequence of characters from among the following:

- The letters `a` through `z` and `A` through `Z`
- The digits `0` through `9`
- The dash (`-`), period (`.`), and underscore (`_`) characters

A MySQL Cluster Manager identifier must begin with a letter or digit.

Case-sensitivity rules for client commands.

The rules for case-sensitivity of MySQL Cluster Manager identifiers, commands, command options, process names, and configuration attributes are as follows:

- *Identifiers are case-sensitive.* For example, `delete site mycluster` cannot be used to delete a site named `myCluster`.
- *Command keywords and the long forms of command options are case-insensitive.* For example, any of the three commands `delete cluster mycluster`, `DELETE CLUSTER mycluster`, and `DeLeTe cLuStEr mycluster` works to delete the MySQL Cluster instance named `mycluster`.

In this manual, we show command keywords and the long forms of command options in lowercase, but you are not required to follow this convention if you do not wish to do so.

- *The short forms of command options are case-sensitive.* For example, `-b` (lowercase) is the short form of the `--basedir` option, but `-B` (uppercase) is the short form of the `--background` option.
- *Names of MySQL Cluster processes are case-insensitive.* For example, either of the commands `get --include-defaults DataMemory:ndbd mycluster` or `get --include-defaults datamemory:NDBD mycluster` reports the data memory allocated for each `ndbd` process in the cluster named `mycluster`.

In this manual, we show names of MySQL Cluster processes in lowercase. You are not required to follow this convention if you do not wish to do so; however, since the corresponding executables are named and must be invoked in lowercase, we suggest that you use lowercase.

- *Configuration attribute names are case-insensitive.* For example, either of the commands `get --include-defaults DataMemory:ndbd mycluster` or `get --include-defaults datamemory:ndbd mycluster` returns the data memory allocated for each `ndbd` process in the cluster named `mycluster`; either of the commands `set engine-condition-pushdown:mysqlld:4=0 mycluster` or `set Engine-Condition-Pushdown:mysqlld:4=0`

`mycluster` disables the condition pushdown optimization in the `mysqld` process having the node ID 4 in the MySQL Cluster named `mycluster`.



Note

Configuration attributes in the MySQL Cluster Manager derive from two different sources: MySQL Cluster configuration parameters, and MySQL Server options. MySQL Cluster configuration parameters are case-insensitive, but their canonical forms use upper camelcase (that is, medial capitalization including the first letter). This means that whether you set a value for data memory using the MySQL Cluster Manager client or in the `config.ini` file, you can refer to it as `DataMemory`, `datamemory`, or `dATAmEMORY` without any negative impact. However, MySQL Server command-line options are case-sensitive and use only lowercase. This means that, for example, `set Engine-Condition-Pushdown:mysqld:4=0 mycluster` in the MySQL Cluster Manager client works to disable condition pushdown in the indicated `mysqld` process, but if you invoke the `mysqld` executable from a system prompt using `--Engine-Condition-Pushdown=0, mysqld` fails to start.

In this manual, for easy recognition, we show configuration attribute names as having the same lettercase used in other MySQL documentation; thus, we always refer to `DataMemory`, rather than `datamemory` or `DATAMEMORY`, and `engine-condition-pushdown`, rather than `Engine-Condition-Pushdown` or `ENGINE-CONDITION-PUSHDOWN`. While you are not required to do this when using MySQL Cluster Manager, we suggest that you also follow this convention.



Note

Values that contain space characters must be quoted using single quote (`'`) characters. For example, if you wish to define a package named `mypackage` for a site named `mysite` using `/usr/local/mysql cluster/7.3` (where a space occurs between `mysql` and `cluster`) as the path to the base directory on all hosts, the correct command would be `add package --basedir='/usr/local/mysql cluster/7.3' mypackage`.

To decrease the possibility of errors in reading and entering MySQL Cluster Manager commands, we recommend avoiding the use of space characters whenever possible.

Each command must end with a terminator character. By default, this is the semicolon (`;`) character. However, the sequences `\g` and `\G` are also supported as command terminators. The `\G` terminator causes the output to be vertically formatted (the same as in the standard `mysql` client), as shown in this example:

```
mcm> get DataMemory mycluster\G
***** 1. row *****
  Name: DataMemory
  Value: 500M
Process1: ndbd
  Id1: 2
Process2:
  Id2:
  Level: Process
  Comment:
***** 2. row *****
  Name: DataMemory
  Value: 500M
Process1: ndbd
  Id1: 3
Process2:
```

```
Id2:
Level: Process
Comment:
2 rows in set (0.22 sec)
```

By convention (for reasons of readability), we do not normally include the command terminator when showing the syntax for a command in Backus-Naur format or when including a MySQL Cluster Manager command inline in this text. However, if you do not use a statement terminator when you enter the command in the MySQL Cluster Manager client, the client displays a special “waiting...” prompt `->` until you supply a terminator, as shown here:

```
mcm> list sites
->
->
->
-> ;
Empty set (1.50 sec)
```

(This is the same as the behavior of the `mysql` client when you fail to end a statement with a terminator.)

A command option can also in many cases accept (or even require) a set of one or more *values*. The next example includes such an option, and also demonstrates setting of multiple values in a single option by passing them to the option as a comma-separated list:

```
mcm> create site --hosts=tonfisk,flundra mysite;
+-----+
| Command result |
+-----+
| Site created successfully |
+-----+
1 row in set (7.41 sec)
```

The command just shown creates a site named `mysite`, consisting of two hosts named `tonfisk` and `flundra`. (See [Section 4.2.6, “The create site Command”](#), for more information about this command.) Since we used the long form of the `--hosts` option, we were required to use an equals sign (=) to mark the end of the option name and the beginning of the values list. You must not insert any space characters before or after the equal sign; doing so causes an error, as shown here:

```
mcm> create site --hosts =grindval,haj yoursite;
ERROR 7 (00MGR): Option --hosts requires a value
mcm> create site --hosts= grindval,haj yoursite;
ERROR 7 (00MGR): Option --hosts requires a value
```

The short form of an option does not use an equal sign. Instead, the value-list is separated from the option by a space. Using the `-h` option, which is the short form of the `--hosts` option, the previous `create site` command can be entered and executed like this:

```
mcm> create site -h tonfisk,flundra mysite;
+-----+
| Command result |
+-----+
| Site created successfully |
+-----+
1 row in set (7.41 sec)
```

The short forms of options actually accept multiple spaces between the option name and the values list; however, a single space is sufficient. If you omit the space, or try to use an equal sign, the command fails with an error, as shown here:

```
mcm> create site -htonfisk,flundra mysite;
ERROR 6 (00MGR): Illegal number of operands
mcm> create site -h=tonfisk,flundra mysite;
ERROR 3 (00MGR): Illegal syntax
```

Any option value containing one or more whitespace characters, one or more dash characters (-), or both, must be quoted using single quotation marks. Multiple values should be separated by commas only; do not insert spaces before or after any of the commas. Using spaces before or after the commas in a list of values causes the command to fail with an error, as shown here:

```
mcm> create site --hosts=tonfisk, flundra mysite;  
ERROR 6 (OOMGR): Illegal number of operands
```

As you can see from the examples just shown, a MySQL Cluster Manager client command returns a result set, just as an SQL statement does in the standard `mysql` client. The result set returned by a MySQL Cluster Manager client command consists of one of the following:

- **A single row that contains a message indicating the outcome of the command.** The `create site` command in the last example returned the result `Site created successfully`, to inform the user that the command succeeded.
- **One or more rows listing requested objects or properties.** An example of such a command is `list processes`, as shown here:

```
mcm> list processes mycluster;  
+-----+-----+-----+  
| NodeId | Name      | Host      |  
+-----+-----+-----+  
| 49     | ndb_mgmd  | flundra   |  
| 1      | ndbd      | tonfisk   |  
| 2      | ndbd      | grindval  |  
| 50     | mysqld    | haj       |  
| 51     | mysqld    | torsk     |  
| 52     | ndbapi    | *         |  
+-----+-----+-----+  
6 rows in set (0.03 sec)
```

In the case of `list processes`, each row in the result contains the ID and type of a node in the MySQL Cluster named `mycluster`, together with the name of the host on which the process is running.

- **An empty result set.** This can occur with one of the `list` commands when there is nothing to report, such as when `list sites` is used before any sites have been created:

```
mcm> list sites;  
Empty set (0.72 sec)
```

Each command must be entered separately; it is not possible to combine multiple commands on a single line.

Options common to client commands.

The following three options are common to most MySQL Cluster Manager client commands:

1. `--help` (short form: `-?`): Common to all client commands. Provides help output specific to the given command. See [Section 4.1, “Online Help and Information Commands”](#), for more information about this option.
2. `--force` (short form `-f`): Causes any safety checks to be bypassed when executing the command. For example, `delete cluster mycluster` normally fails if any of the MySQL Cluster processes in the MySQL Cluster named `mycluster` are running; however, `delete cluster --force mycluster` forces the shutdown of `mycluster`, followed by the deletion of `mycluster` from MySQL Cluster Manager's inventory.

The `--force` option is supported for the following MySQL Cluster Manager client commands:

- `delete site`

- `start cluster`
 - `restart cluster`
 - `stop cluster`
 - `delete cluster`
 - `add process`
 - `start process`
 - `stop process`
 - `remove process`
 - `set`
 - `reset`
3. `--background` (short form `-B`): Rather than waiting for the command to complete, the MySQL Cluster Manager client immediately returns the command prompt, allowing you to perform additional tasks in the client while that command continues to execute in the background. This can be useful when executing commands that might require some time to complete (such as starting a cluster with a great many nodes).

This option is supported by all client commands except for `create site`, `delete site`, `add hosts`, `add package`, and `delete package`.

4.1 Online Help and Information Commands

Online help is available in the MySQL Cluster Manager client for MySQL Cluster Manager client commands. The client can provide both general and command-specific information. In addition, you can obtain information about `mysql` client commands that are independent of the MySQL server and thus are also available for use when connected to the MySQL Cluster Manager agent.

Listing MySQL Cluster Manager client commands.

For a list of all commands with brief descriptions, use the `list commands` command, as shown here:

```
mcm> list commands;
```

Help	
COMMANDS	
<code>abort backup</code>	Abort an ongoing cluster backup.
<code>add hosts</code>	Add hosts to site.
<code>add package</code>	Add a package alias.
<code>add process</code>	Add cluster process.
<code>backup cluster</code>	Backup a cluster.
<code>change log-level</code>	Change the log-level
<code>change process</code>	Change process type.
<code>collect logs</code>	Collect log files.
<code>create cluster</code>	Create a cluster.
<code>create site</code>	Create a site.
<code>delete cluster</code>	Delete a cluster.
<code>delete package</code>	Delete a package.
<code>delete site</code>	Delete a site.
<code>get</code>	Get configuration variables.
<code>import cluster</code>	Import a running cluster.

```

| import config      Import the configuration of a running cluster.
| list backups      List backup images.
| list clusters     List all clusters.
| list commands     List the help text.
| list hosts        List hosts in site.
| list nextnodeids  List next nodeids to be allocated.
| list packages     List all packages.
| list processes    List processes.
| list sites        List all sites.
| remove hosts      Remove hosts from site.
| remove process    Remove a cluster process.
| reset            Reset configuration variables.
| restart cluster   Restart a cluster.
| restore cluster   Restore a cluster.
| rotate log        Rotate the mcmd log.
| set              Set configuration variables.
| show settings     Show agent settings.
| show status       Show cluster, process, operation or backup status.
| start cluster     Start a cluster.
| start process     Start a cluster process.
| stop agents       Stop agents in site.
| stop cluster      Stop a cluster.
| stop process      Stop a cluster process.
| upgrade cluster   Upgrade a cluster.
| version           Print version information.

GLOBAL OPTIONS
Options that can be used with all commands

--help|-?          Print detailed help.

Use '<COMMAND> --help' to see verbose help for individual commands.
-----+
49 rows in set (0.04 sec)

```

Obtaining information about specific MySQL Cluster Manager client commands.

To obtain more detailed help specific to a given command, invoke the command using the `--help` option, as shown in this example:

```

mcm> create site --help;
-----+
| Help
-----+
|
| create site [options] <sitename>
|
| Creates a site from the hosts listed in --hosts.
|
| Required options:
| --hosts|-h          Comma separated list of hostnames.
|                    Format: --hosts = <host>[,<host>]*.
|
| Valid options:
| --background|-B    Run command in background, and return prompt
|                    to the user immediately.
| --hosts|-h          Comma separated list of hostnames.
|                    Format: --hosts = <host>[,<host>]*.
|
-----+
13 rows in set (0.00 sec)

```

For any MySQL Cluster Manager client command, the `--help` option may be abbreviated to `-?`:

```

mcm> list processes -?;
-----+
| Help
-----+

```

```
list processes <sitename>

Lists all processes defined in the specified cluster.

-----+
4 rows in set (0.00 sec)
```

As mentioned elsewhere in this manual (see [Chapter 4, MySQL Cluster Manager Client Commands](#)), many other MySQL Cluster Manager command options have short forms as well. These are included in the documentation for each command. You can also find out what these are for a given command by invoking it with the `--help` or `-?` option.

You can obtain the release version of the MySQL Cluster Manager software in use from the output of the `version` command.

mysql client commands in the MySQL Cluster Manager client.

You can also use most standard `mysql` client commands in the MySQL Cluster Manager client (but *not* SQL statements, which depend on being connected to a MySQL server), such as `prompt`, `quit`, and `status`. For example, the output of the `status` command when connected to the MySQL Cluster Manager agent looks something like this (depending on the exact version of the client and agent which you are using and possibly other factors):

```
mcm> status
-----
/home/jon/bin/mcm/libexec/./cluster/bin/mysql
Ver 14.14 Distrib 5.6.24-ndb-7.4.6, for linux2.6 (x86_64) using EditLine wrapper

Connection id:          1
Current database:      <n/a>
Current user:          admin
SSL:                   Not in use
Current pager:         less
Using outfile:         ''
Using delimiter:       ;
Server version:        1.3.6 MySQL Cluster Manager
Protocol version:      10
Connection:            127.0.0.1 via TCP/IP
Server characterset:   <n/a>
Db characterset:       <n/a>
Client characterset:   <n/a>
Conn. characterset:    <n/a>
TCP port:              1862
-----
```



Note

You may use the command delimiter with `mysql` client commands, but you are not required to do so. For instance, assuming that the delimiter in use was the default semicolon (;) character, we could have executed the `status` command like this:

```
mcm> status;
-----
/home/jon/bin/mcm/cluster/bin/mysql Ver 14.14 Distrib 5.6.24-ndb-7.4.6,...
```

A particularly useful `mysql` client command that you can also employ with `mcm` is the `source` command (short form: `\.`), which you can use for executing scripts containing MySQL Cluster Manager client commands. On a Linux system, you might have a text file in your home directory named `get-attributes.mcm`, whose contents are shown here:

```
get :ndb_mgmd mycluster\G
get :ndbd mycluster\G
get :mysqld mycluster\G
```

Assuming that you have created a cluster named `mycluster`, you can run this script in the client; the results vary according to how this cluster is actually configured, but should be similar to this:

```
mcm> \. ~/get-attributes.mcm
mcm> get :ndb_mgmd mycluster\G
***** 1. row *****
  Name: DataDir
  Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/49/data
Process1: ndb_mgmd
  NodeId1: 49
Process2:
  NodeId2:
  Level:
  Comment:
***** 2. row *****
  Name: HostName
  Value: flundra
Process1: ndb_mgmd
  NodeId1: 49
Process2:
  NodeId2:
  Level:
  Comment: Read only
***** 3. row *****
  Name: NodeId
  Value: 49
Process1: ndb_mgmd
  NodeId1: 49
Process2:
  NodeId2:
  Level:
  Comment: Read only
***** 4. row *****
  Name: PortNumber
  Value: 1186
Process1: ndb_mgmd
  NodeId1: 49
Process2:
  NodeId2:
  Level: Process
  Comment:
4 rows in set (0.09 sec)

mcm> get :ndbd mycluster\G
***** 1. row *****
  Name: DataDir
  Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/1/data
Process1: ndbd
  NodeId1: 1
Process2:
  NodeId2:
  Level:
  Comment:
***** 2. row *****
  Name: HostName
  Value: tonfisk
Process1: ndbd
  NodeId1: 1
Process2:
  NodeId2:
  Level:
  Comment: Read only
***** 3. row *****
  Name: NodeId
  Value: 1
Process1: ndbd
  NodeId1: 1
```



```

Process2:
NodeId2:
Level:
Comment: Read only
***** 4. row *****
Name: DataDir
Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/2/data
Process1: ndbd
NodeId1: 2
Process2:
NodeId2:
Level:
Comment:
***** 5. row *****
Name: HostName
Value: grindval
Process1: ndbd
NodeId1: 2
Process2:
NodeId2:
Level:
Comment: Read only
***** 6. row *****
Name: NodeId
Value: 2
Process1: ndbd
NodeId1: 2
Process2:
NodeId2:
Level:
Comment: Read only
6 rows in set (0.10 sec)

mcm> get :mysqld mycluster\G
***** 1. row *****
Name: datadir
Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/50/data
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment:
***** 2. row *****
Name: HostName
Value: haj
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment: Read only
***** 3. row *****
Name: log_error
Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/50/data/mysqld_50_out.err
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment:
***** 4. row *****
Name: ndb_nodeid
Value: 50
Process1: mysqld
NodeId1: 50
Process2:

```

```

NodeId2:
Level:
Comment: Read only
***** 5. row *****
Name: ndbcluster
Value:
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment: Read only
***** 6. row *****
Name: NodeId
Value: 50
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment: Read only
***** 7. row *****
Name: port
Value: 3306
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment:
***** 8. row *****
Name: socket
Value: /tmp/mysql.mycluster.50.sock
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment:
***** 9. row *****
Name: tmpdir
Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/50/data/tmp
Process1: mysqld
NodeId1: 50
Process2:
NodeId2:
Level:
Comment:
***** 10. row *****
Name: datadir
Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/51/data
Process1: mysqld
NodeId1: 51
Process2:
NodeId2:
Level:
Comment:
***** 11. row *****
Name: HostName
Value: torsk
Process1: mysqld
NodeId1: 51
Process2:
NodeId2:
Level:
Comment: Read only
***** 12. row *****

```

```

    Name: log_error
    Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/51/data/mysqld_51_out.err
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment:
***** 13. row *****
    Name: ndb_nodeid
    Value: 51
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment: Read only
***** 14. row *****
    Name: ndbcluster
    Value:
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment: Read only
***** 15. row *****
    Name: NodeId
    Value: 51
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment: Read only
***** 16. row *****
    Name: port
    Value: 3307
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment:
***** 17. row *****
    Name: socket
    Value: /tmp/mysql.mycluster.51.sock
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment:
***** 18. row *****
    Name: tmpdir
    Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/51/data/tmp
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment:
18 rows in set (0.05 sec)

mcm>

```

**Note**

You are not returned to the client prompt until the script has finished executing.

Similarly, on Windows, you can create a batch file using Notepad or another text editor, copy the same `get` commands as shown previously into it, and save it as `get-attributes.bat` in a convenient location such as the Windows desktop.

You can view a list of available `mysql` client commands using the `help` command. For more information about these, view the `help` output or see [mysql Client Commands](#), in the *MySQL Manual*.

4.2 MySQL Cluster Manager Site and Agent Commands

In this section, we discuss commands used to work with MySQL Cluster Manager management sites. In addition, the `stop agents`, `show settings`, `version`, and `show warnings` commands, which relate to management agents, are also covered in this section.

A *site*, in terms of MySQL Cluster and MySQL Cluster Manager, is a collection of one or more host computers where MySQL Cluster Manager agents are running. Each agent is identified by the combination of two pieces of information:

- The hostname or IP address of the machine where the agent is running
- The number of the port used by the agent for communications

**Note**

MySQL Cluster makes extremely intensive use of network connections, and DNS lookups can contend with MySQL Cluster and MySQL Cluster Manager for bandwidth, resulting in a negative impact on the performance of MySQL Cluster and the applications using it. For this reason, we recommend that you use numeric IP addresses rather than hostnames for MySQL Cluster and MySQL Cluster Manager host computers whenever feasible.

4.2.1 The `add hosts` Command

```
add hosts --hosts=host_list site_name

host_list:
  host[, host[, ...]]
```

This command adds one or more hosts to an existing management site. Agents using the same port as the management site must be running on any hosts added using this command. This command takes two mandatory arguments: a list of hosts (using the `--hosts` option), and the name of the site to which the hosts are to be added.

The `--hosts` takes a comma-separated list of one or more hosts to be added to the site.

For example, the following command adds two hosts named `torsk` and `kolja` to management site `mysite`:

```
mcm> add hosts --hosts=torsk,kolja mysite;
+-----+
| Command result          |
+-----+
| Hosts added successfully |
+-----+
```

```
1 row in set (0.48 sec)
```

None of the hosts added by this command may already be members of management site `site_name`. Do not attempt to add again a host that is already a member of the management site using its secondary (or alternate) IP address—the `mcmd` process on the host is already bound to the IP address that was supplied when the host was first added, and it cannot be bound again to another IP address.



Notes

- This command does not support the `--force` option.
- Do not use `localhost` in the host list, as MySQL Cluster Manager relies on the operating system for host name resolution, and `localhost` might be resolved differently on different systems. Use proper host names for the host list or, preferably, use the IP addresses for the hosts instead.
- When IPv6-enabled Windows systems are used as MySQL Cluster hosts under MySQL Cluster Manager, you must reference these hosts using IPv4 addresses. Otherwise, MySQL Cluster Manager will be unable to connect to the agent processes on those hosts. See [Section 5.1, “MySQL Cluster Manager Usage and Design Limitations”](#).

4.2.2 The `remove hosts` Command

```
remove hosts --hosts=host_list site_name
```

```
host_list:
  host[, host[, ...]]
```

This command removes one or more hosts from an existing management site. It takes as arguments a required option `--hosts`, whose value is a comma-separated list of one or more hosts to be removed, and the name of the site from which the hosts are to be removed. A number of limitations apply:

- The name of the host to be removed must not be `localhost` or `127.0.0.1`.
- The host to be removed must not have any managed processes from any clusters assigned to them (remove those processes first with the `remove process` command); it can have unmanaged processes assigned to them though (typically, `ndbapi@hostname` or `mysqld@*hostname`).
- There should not be any packages defined with explicit paths pointing to the host to be removed.
- A quorum consists of a majority of hosts (i.e., half of the total number of hosts plus one) must exist for the site both before and after the host's removal, or it will not be possible to execute the `remove host` command.
- You cannot remove the last host from a site; use the `delete site` command instead.

The following command removes two hosts named `tonfisk` and `flundra` from the management site `mysite`:

```
mcm> remove hosts --hosts=tonfisk,flundra mysite;
```

```
+-----+
| Command result |
+-----+
| Hosts removed successfully |
+-----+
1 row in set (0.48 sec)
```

4.2.3 The `change log-level` Command

```
change log-level [--hosts=|-h ]host_list] log_level site_name

host_list:
    host[,host[,...]]
```

Set the management agent's cluster logging level. This has the same effect as using the `--log-level` option; however, unlike the option, this command can be used at run time and does not require a restart of `mcmd`. Issuing this command overrides any value for `--log-level` set on the command line or in the agent configuration file.

When used with the `log_level` alone without a `host_list` and a `site_name`, this command applies only to the agent the `mcm` client is connected to. In the following example, the logging level is set to `warning` only on the host managed directly by the agent to which the `mcm` client is connected:

```
mcm> change log-level warning;
+-----+
| Command result |
+-----+
| Log-level changed successfully |
+-----+
1 row in set (0.00 sec)
```

You can specify the name of a site to be affected by the command. For example, the following invocation of the command applies to the site named `mysite`:

```
mcm> change log-level debug mysite;
+-----+
| Command result |
+-----+
| Log-level changed successfully |
+-----+
1 row in set (0.05 sec)
```

You can also restrict the change to one or more hosts in a given site using the `--hosts` option, with multiple host names separated by commas. The following command changes the logging level to debug on the hosts named `tonfisk` and `haj`, but not on any other hosts in `mysite`:

```
mcm> change log-level --hosts=tonfisk,haj debug mysite;
+-----+
| Command result |
+-----+
| Log-level changed successfully |
+-----+
1 row in set (0.09 sec)
```

You must specify a site when using the `--hosts` option; trying to use `--hosts` alone results in an error.

Accepted values for `log_level` are the same as for the `--log-level`: one of `debug`, `critical`, `error`, `info`, `message`, or `warning`. For more detailed information about the meanings and effects of these values, see [NDB Cluster Logging Management Commands](#).

4.2.4 The `rotate log` Command

```
rotate log [--hosts=host_list] [site_name]

host_list:
    host[,host[,...]]
```

Rotate `mcmd` logs for the connected MySQL Cluster Manager agent, for agents running on certain hosts, or for agents on all hosts in a management site.

For example, to rotate logs for the agent to which the client session is connected:

```
mcm> rotate log;
+-----+
| Command result          |
+-----+
| Log rotated successfully |
+-----+
1 row in set (0.03 sec)
```

A new log file, with a timestamp in its name, is created as a result:

```
-rw-r----- 1 mcmd cluster 74265 Jul 15 22:45 mcmd.log
-rw-r----- 1 mcmd cluster 1197573 Jul 15 22:45 mcmd.log.2014-07-15T22-45-28
```

To rotate logs for agents on hosts `nanna12` and `nanna13`:

```
mcm> rotate log --hosts=nanna12,nanna13 mysite;
```

To rotate logs on all agents in the management site `mysite`:

```
mcm> rotate log mysite;
```

4.2.5 The `collect logs` Command

```
collect logs [cluster_name]
```

This command collects log files and other related files from all hosts. When the name of a cluster (*cluster_name*) is supplied with the command, it collects all the log files (`.log`) as well as the configuration files (`.ini`, `.cnf`), error files (`.err`), and option files (`.opt`) used by all processes belonging to the cluster, and also all the agent log files.

When an `mcmd` agent receives the `collect logs` command from the `mcm` agent it is connected with, it sets up a TCP server socket using port 0, and lets the operating system assign the actual port number. All agents in the site are then instructed to perform the copying, and each of them spawns a TCP client, which connects to the TCP server socket set up earlier to copy the files over.



Warning

If a firewall or other networking issues prohibit the TCP clients to connect to the TCP server socket, the `collect logs` command will never complete.

The collected files are put under the MySQL Cluster Manager data repository (`./mcm_data` (relative to the installation directory of MySQL Cluster Manager) by default, or specified by the option `--manager-directory`) inside a folder named `collected_files`, under which the files are organized under a hierarchy of timestamps [for the file collections] and then host names; below the host name level, the file organization mimics that on the host. For example, the agent log for the host `tonfisk`, collected on 2014-07-31 at `07:44:05`, is found at:

```
/opt/mcm_data/collected-files/2014-07-31T07:44:05Z/tonfisk/opt/mysql/logs/mcmd-tonfisk-19001.log
```

If *cluster_name* is omitted, only the agent log files are collected.

4.2.6 The `create site` Command

```
create site {--hosts=|-h }host_list site_name
host_list:
```

```
host[,host[,...]]
```

The `create site` command is used to create a MySQL Cluster Manager management site; that is, a set of MySQL Cluster Manager management agents running on one or more host computers. The command requires a list of one or more hosts where management agents are running and a name for the site. The host list is passed as the value of the `--hosts` option (short form: `-h`).

This is an example of a `create site` command that creates a site named `mysite`, consisting of the hosts `tonfisk` and `flundra`:

```
mcm> create site --hosts=tonfisk,flundra mysite;
+-----+
| Command result |
+-----+
| Site created successfully |
+-----+
1 row in set (0.31 sec)
```



Tip

You can verify that the site was created as intended, using the `list sites` command, as shown here:

```
mcm> list sites;
+-----+-----+-----+-----+
| Site | Port | Local | Hosts |
+-----+-----+-----+-----+
| mysite | 1862 | Local | tonfisk,flundra |
+-----+-----+-----+-----+
1 row in set (0.06 sec)
```

(See [Section 4.2.8, “The list sites Command”](#), for more information about this command.)

Agents must be running on all hosts specified in the `--hosts` option when `create site` is executed; otherwise, the command fails with the error `Agent on host host:port is unavailable`. The host where the agent used to issue the command is running must be one of the hosts listed. Otherwise, the command fails with the error `Host host_name is not a member of site site_name`.

A given agent may be a member of one site only; if one of the management agents specified in the `host_list` already belongs to a site, the command fails with the error `Host host is already a member of site site`.



Notes

- Do not use `localhost` in the host list, as MySQL Cluster Manager relies on the operating system for host name resolution, and `localhost` might be resolved differently on different systems. Use proper host names for the host list or, preferably, use the IP addresses for the hosts instead.
- When IPv6-enabled Windows systems are used as MySQL Cluster hosts under MySQL Cluster Manager, you must reference these hosts using IPv4 addresses. Otherwise, MySQL Cluster Manager will be unable to connect to the agent processes on those hosts. See [Section 5.1, “MySQL Cluster Manager Usage and Design Limitations”](#).

4.2.7 The `delete site` Command


```
delete site site_name
```

The `delete site` command deletes an existing management site. The command does not stop or remove any agents making up the deleted site; instead, these agents continue to run, and remain available for use in other sites.

The command takes a single argument, the name of the site to be deleted. This example shows the deletion of a management site named `mysite`:

```
mcm> delete site mysite;
+-----+
| Command result          |
+-----+
| Site deleted successfully |
+-----+
1 row in set (0.38 sec)
```

If the site to be deleted does not exist, the command fails with the error `Command requires a site to be defined`. If there are any packages referencing hosts belonging to the site, `delete site` fails with the error `Packages exist in site site_name`. The command also fails if there are defined any clusters that include hosts belonging to the site.



Note

The management client must be connected to a site in order to be able to delete it.

In addition, if you execute a `delete site` command with the `--force` option using one management agent while a different management agent is not running, you must remove the “missing” management agent's site files manually. For more information on site files, see [Section 2.4, “MySQL Cluster Manager Configuration File”](#).

4.2.8 The `list sites` Command

```
list sites
```

This command returns a list of the sites known to the management agent. It does not require any arguments. An example is shown here:

```
mcm> list sites;
+-----+-----+-----+-----+
| Site  | Port | Local | Hosts          |
+-----+-----+-----+-----+
| mysite | 1862 | Local | tonfisk,flundra |
+-----+-----+-----+-----+
1 row in set (0.06 sec)
```

The output of `list sites` contains the following columns:

- **Site.** The name of the site.
- **Port.** The TCP/IP port used for communications between clients and management agents.
- **Local.** Either one of `Local` or `Remote`.
- **Hosts.** A comma-separated list of the hosts making up the site.

4.2.9 The `list hosts` Command

```
list hosts site_name
```

The `list hosts` command is used to obtain a list of the hosts comprising a given management site. The command requires a single argument, the name of the site to be examined. For each host listed, the information returned includes the hostname, status, and version of the management agent software in use, as shown in this example:

```
mcm> list hosts mysite;
+-----+-----+-----+
| Host      | Status  | Version |
+-----+-----+-----+
| tonfisk   | Available | 1.3.6   |
+-----+-----+-----+
| flundra   | Available | 1.3.6   |
+-----+-----+-----+
2 rows in set (0.16 sec)
```

If you omit the `site_name` argument, the command fails with an error, as shown here:

```
mcm> list hosts;
ERROR 6 (00MGR): Illegal number of operands
```

Using the `--verbose` option (short form : `-v`) causes the command to print additional information on the hosts:

```
mcm> list hosts --verbose mysite;
+-----+-----+-----+-----+-----+-----+
| Host      | Status  | Version | Cores | Memory | OS
+-----+-----+-----+-----+-----+-----+
| tonfisk   | Available | 1.3.6   | 1      | 1819 Mb | Linux 3.13.11-100.fc19.x86_64
| flundra   | Available | 1.3.6   | 1      | 1819 Mb | Linux 3.13.11-100.fc19.x86_64
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.07 sec)
```

4.2.10 The `show settings` Command

```
show settings
```

This command lists the current values of a number of `mcmd` options:

```
mcm> show settings;
+-----+-----+
| Setting          | Value
+-----+-----+
| log-file         | /opt/mcm_data/mcmd.log
| log-level        | message
| log-use-syslog   | FALSE
| manager-directory | /opt/mcm_data
| manager-username | mcmd
| manager-password | *****
| manager-port     | 1862
| xcom-port        | 18620
+-----+-----+
8 rows in set (0.00 sec)
```

The `show settings` command takes no arguments.

4.2.11 The `stop agents` Command

```
stop agents [--hosts=host_list] site_name
```

This command stops one or more MySQL Cluster Manager agents on one or more hosts.

When used without any arguments, `stop agents` stops the agent to which the client is currently connected.

When used with the name of a management site, the command stops all agents running on hosts making up the site. The following stops all MySQL Cluster Manager agents running on hosts in `mysite`:

```
mcm> stop agents mysite;
```

You can also stop a subset of the agents in a given management site by listing the hosts where they are running with the `--hosts` option, along with the name of the site to which they belong. The result of the following command is to stop MySQL Cluster Manager agents running on hosts `kolja` and `torsk`, both of which are members of the management site `mysite`:

```
mcm> stop agents --hosts=kolja,torsk mysite;
```

Multiple host names following the `--hosts` option should be separated by commas, with no intervening spaces. Invoking `stop agents` with this option without supplying a `site_name` causes a syntax error. Using an undefined `site_name` or names of hosts not belonging to the site with this command also results in an error.



Note

When IPv6-enabled Windows systems are used as MySQL Cluster hosts under MySQL Cluster Manager, you must reference these hosts using IPv4 addresses. Otherwise, MySQL Cluster Manager will be unable to connect to the agent processes on those hosts. See [Section 5.1, “MySQL Cluster Manager Usage and Design Limitations”](#).

4.2.12 The `version` Command

```
version
```

This command displays the version of the MySQL Cluster Manager software in use by the MySQL Cluster Manager agent to which this client is connected, as shown here:

```
mcm> version;
+-----+
| Version |
+-----+
| MySQL Cluster Manager 1.3.6 (64bit) |
+-----+
1 row in set (0.00 sec)
```

The `version` command takes no arguments.

4.2.13 The `show warnings` Command

Using the `show warnings` command (available for MySQL Cluster Manager 1.3.4 and after), you can check the warnings (up to the last five) issued to the agent log (`mcmd.log`). For example:

```
mcm> set delayed_insert_timeout:mysql=400 mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+

mcm> show warnings;
+-----+-----+-----+
| Level | Code | Message |
+-----+-----+-----+
| Warning | -1 | Config variable delayed_insert_timeout was deprecated in mysqld 5.6.7 |
+-----+-----+-----+
```

4.3 MySQL Cluster Manager Package Commands

This section contains information about MySQL Cluster Manager client commands used to register, extend, unregister, and obtain information about the software packages making up instances of MySQL Cluster that are to be managed using the MySQL Cluster Manager.

4.3.1 The `add package` Command

```
add package {--basedir=|-b }path
            [--hosts=|-h ]host_list package_name

host_list:
    host[,host[,...]]
```

This command creates a new package, or, if the package named `package_name` already exists, this command extends the package definition.

The `--basedir` option (short form: `-b`) indicates the location of the MySQL Cluster installation directory on the listed hosts, and is required. This must be the path to the top-level directory where the MySQL Cluster software is located (for example, `/usr/local/mysql`), and should *not* include the MySQL Cluster `bin`, `libexec`, or other subdirectory within the installation directory.

Hosts may be specified as a comma-separated list, using the `--hosts` option (short form: `-h`); however, this option is not required. If `--hosts` is omitted, the `path` is assumed to be valid for all hosts in the cluster that is created using this package (see [Section 4.4.1, “The create cluster Command”](#)).



Important

- You cannot perform `add package` if you have not yet defined any sites (each host referenced in an `add package` command must be associated with a site). See [Section 4.2.6, “The create site Command”](#), for more information about defining sites.
- When a package is first added for a site with the `add package` command, whenever the `--hosts` option is used, the host list must contain the host for the `mcmd` agent to which the `mcm` client is currently connected, in order to allow the MySQL Cluster Manager to access the version information of the package.

Suppose we have two Linux hosts named `tonfisk` and `flundra`, and the MySQL Cluster software is installed in `/usr/local/mysql` on both hosts. In this case, you can create a package named `mypackage` that accounts for both hosts as shown here:

```
mcm> add package --basedir=/usr/local/mysql mypackage;
+-----+
| Command result |
+-----+
| Package added successfully |
+-----+
1 row in set (0.71 sec)
```

When this package is used to create a cluster, the MySQL Cluster Manager knows that it should find the MySQL Cluster software in the `/usr/local/mysql` directory on each of the hosts.

For options to MySQL Cluster Manager client command options having Windows paths as values, you must use forward slashes (/) in place of backslashes (\), so if `tonfisk` and `flundra` are Windows hosts where MySQL Cluster has been installed to the directory `C:\mysql`, the corresponding `add package` command would look like this (with the `--basedir` option highlighted):

```
mcm> add package --basedir=c:/mysql mypackage;
```

The `add package` Command

```
+-----+
| Command result |
+-----+
| Package added successfully |
+-----+
1 row in set (0.71 sec)
```

In the example just given, we could also have issued the command as `add package --basedir=/usr/local/mysql --hosts=tonfisk,flundra mypackage` (or `add package --basedir=c:/mysql --hosts=tonfisk,flundra mypackage` on Windows) with the same result, but the `--hosts` option was not required, since the MySQL Cluster software's location is the same on each host. Let us suppose, however, that the software is installed in `/usr/local/ndb-host-10` on host `tonfisk` and in `/usr/local/ndb-host-20` on host `flundra`. In this case, we must issue 2 separate commands, specifying the host as well as the base directory in each case, as shown here:

```
mcm> add package --basedir=/usr/local/ndb-host-10
> --hosts=tonfisk yourpackage;
+-----+
| Command result |
+-----+
| Package added successfully |
+-----+
1 row in set (0.68 sec)

mcm> add package --basedir=/usr/local/ndb-host-20
> --hosts=flundra yourpackage;
+-----+
| Command result |
+-----+
| Package added successfully |
+-----+
1 row in set (0.81 sec)
```

Assuming that both hosts belong to a site called `mysite`, you can verify that these packages have been created as desired using the `list packages` command, as shown here:

```
mcm> list packages mysite;
+-----+-----+-----+
| Package | Path | Hosts |
+-----+-----+-----+
| yourpackage | /usr/local/ndb-host-10 | tonfisk |
| | /usr/local/ndb-host-20 | flundra |
| mypackage | /usr/local/mysql | tonfisk,flundra |
+-----+-----+-----+
3 rows in set (1.07 sec)
```

(For more information about this command, see [Section 4.3.3, "The `list packages` Command"](#).)

It is possible to assign the same base directory (or directories) on the same host (or hosts) to multiple packages, as shown in this example, in which we assume that hosts `tonfisk` and `flundra` have previously been assigned to a site named `mysite`:

```
mcm> add package -b /usr/local/mysql-cluster mypackage;
+-----+
| Command result |
+-----+
| Package added successfully |
+-----+
1 row in set (1.41 sec)

mcm> add package -b /usr/local/mysql-cluster yourpackage;
+-----+
| Command result |
```

```

+-----+
| Package added successfully |
+-----+
1 row in set (1.58 sec)

mcm> list packages mysite;
+-----+-----+-----+
| Package | Path | Hosts |
+-----+-----+-----+
| mypackage | /usr/local/mysql-cluster | tonfisk,flundra |
| yourpackage | /usr/local/mysql-cluster | tonfisk,flundra |
+-----+-----+-----+
2 rows in set (0.50 sec)

```

**Note**

When IPv6-enabled Windows systems are used as MySQL Cluster hosts under MySQL Cluster Manager, you must reference these hosts using IPv4 addresses. Otherwise, MySQL Cluster Manager will be unable to connect to the agent processes on those hosts. See [Section 5.1, “MySQL Cluster Manager Usage and Design Limitations”](#).

4.3.2 The `delete package` Command

```

delete package [--hosts=-h ]host_list package_name

host_list:
    host[,host[,...]]

```

This command is used to unregister a package. More specifically, it removes any references to MySQL Cluster software installations added to the agent's repository when the package was created. `delete package` does *not* remove any MySQL Cluster installations; the command removes only references to the installations. Once a package has been unregistered, it can no longer be used for a `create cluster` command. The MySQL Cluster binaries remain, but cannot be used in a MySQL Cluster administered using the MySQL Cluster Manager unless and until the base directory containing them has been registered with another package. (It is possible to register a base directory with multiple packages; see [Section 4.3.1, “The `add package` Command”](#), for more information and an example.)

If the `--hosts` option (short form: `-h`) is used with this command, the base directory settings for the host or hosts named by the option are removed as well. All hosts given in the `host_list` must be members of the site to which the package is registered. Otherwise, the command fails.

A package that is in use by a cluster cannot be unregistered; the cluster must first be deleted (see [Section 4.4.2, “The `delete cluster` Command”](#)).

Here is an example that demonstrates how to unregister a package named `mypackage`:

```

mcm> delete package mypackage;
+-----+-----+-----+
| Command result |
+-----+-----+-----+
| Package deleted successfully |
+-----+-----+-----+
1 row in set (1.23 sec)

```

You can also verify that the package was unregistered using the `list packages` command; the package name should no longer appear in the output of this command. If you attempt to use the unregistered package in a `create cluster` command, the command fails, as shown here:

```

mcm> create cluster --package=mypackage
> --processhosts=ndb_mgmd@tonfisk,ndbd@grindval,ndbd@flundra,mysqld@tonfisk mycluster;

```

```
ERROR 4001 (00MGR): Package mypackage not defined
```

An `upgrade cluster` command that references an unregistered package also fails.



Note

When IPv6-enabled Windows systems are used as MySQL Cluster hosts under MySQL Cluster Manager, you must reference these hosts using IPv4 addresses. Otherwise, MySQL Cluster Manager will be unable to connect to the agent processes on those hosts. See [Section 5.1, “MySQL Cluster Manager Usage and Design Limitations”](#).

4.3.3 The `list packages` Command

```
list packages [package_name] site_name
```

This command lists registered packages. It requires a single argument, that being the name of the site with which the packages are registered, as shown in this example:

```
mcm> list packages mysite;
+-----+-----+-----+
| Package | Path | Hosts |
+-----+-----+-----+
| yourpackage | /usr/local/ndb-host-10 | tonfisk |
| | /usr/local/ndb-host-20 | flundra |
| mypackage | /usr/local/mysql | tonfisk,flundra |
+-----+-----+-----+
3 rows in set (1.07 sec)
```

If `tonfisk` and `flundra` are Windows hosts, the list of packages might look something like this:

```
mcm> list packages mysite;
+-----+-----+-----+
| Package | Path | Hosts |
+-----+-----+-----+
| yourpackage | c:/cluster/ndb-host-10 | tonfisk |
| | c:/cluster/ndb-host-20 | flundra |
| mypackage | c:/mysql | tonfisk,flundra |
+-----+-----+-----+
3 rows in set (1.07 sec)
```

In the example just shown, `yourpackage` uses the MySQL Cluster binaries installed at `C:\cluster\ndb-host-10` on host `tonfisk`, and at `C:\cluster\ndb-host-20` on `flundra`; `mypackage` uses MySQL Cluster binaries installed at `C:\mysql` on both hosts.

The output contains three columns; these are described in the following list:

- **Package.** The name of the package. This can sometimes be empty when a package includes MySQL Cluster installations that are in different locations on different hosts (see next example).
- **Path.** The path to the MySQL Cluster installation directory (base directory) on the indicated host or hosts. This is the same as the value given for the `--basedir` option in the `add package` command that was used to create or augment the package.

On Windows, paths shown in this column have any backslash characters converted to forward slashes, just as must be done for the `--basedir` option (see the earlier example in this section).

- **Hosts.** The host or hosts where the MySQL Cluster installation or installations are located.

You can filter the results so that information relating to only a single package is displayed by supplying the package name before the site name, as shown here:

```
mcm> list packages yourpackage mysite;
+-----+-----+-----+
| Package | Path | Hosts |
+-----+-----+-----+
| yourpackage | /usr/local/ndb-host-10 | tonfisk |
| | /usr/local/ndb-host-20 | flundra |
+-----+-----+-----+
2 rows in set (0.55 sec)
```

(See [Section 4.3.1, “The add package Command”](#), for the `add package` commands that were used to create `yourpackage`.)

When a package contains MySQL Cluster installations using different base directories on different hosts, each unique combination of path and host is shown in its own row. However, the name of the package is displayed in the first row only; all rows that immediately follow this row and that do not contain the package name also relate to the same package whose name is shown in the first preceding row to display a package name. For example, consider the `list packages` command and output shown here:

```
mcm> list packages mysite;
+-----+-----+-----+
| Package | Path | Hosts |
+-----+-----+-----+
| yourpackage | /usr/local/ndb-host-10 | tonfisk |
| | /usr/local/ndb-host-20 | flundra |
| mypackage | /usr/local/mysql | tonfisk |
| | /usr/local/bin/mysql | flundra |
+-----+-----+-----+
3 rows in set (1.07 sec)
```

This output shows that there are two packages defined for the site named `mysite`; these packages are named `yourpackage` and `mypackage`. The package `yourpackage` consists of the MySQL Cluster binaries in the directory `/usr/local/ndb-host-10` on host `tonfisk`, and in the directory `/usr/local/ndb-host-20` on host `flundra`. The package named `mypackage` consists of the MySQL Cluster binaries in the directory `/usr/local/mysql` on host `tonfisk`, and in the directory `/usr/local/bin/mysql` on host `flundra`.

If you omit the `site_name` argument, the command fails with an error, as shown here:

```
mcm> list packages;
ERROR 6 (00MGR): Illegal number of operands
```

4.4 MySQL Cluster Manager Cluster Commands

This section contains descriptions of MySQL Cluster Manager commands used to perform operations on clusters. These include creating and deleting a cluster; starting, stopping, and restarting a cluster; upgrading a cluster (that is, upgrading the MySQL Cluster software used by a given cluster); and listing clusters known to MySQL Cluster Manager.

4.4.1 The `create cluster` Command

```
create cluster [--package=-P ]package_name
  [--processhosts=-R ]process_host_list cluster_name
  [(--import|-m) cluster_name] [--verbose | -v]

process_host_list:
  process_name[:node_id]@host[,process_name@host[,...]]

process_name:
  {ndb_mgmd|ndbd|ndbmtdd|mysqld|ndbapi}
```


This command creates a cluster to be managed by the MySQL Cluster Manager. However, it does not start the cluster (see [Section 4.4.7, “The `start cluster` Command”](#)).

This command can also be used to create a cluster earmarked specifically as a target for importing another cluster that is not already under MySQL Cluster Manager control, as described later in this section, by employing the `--import` option. See also [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#).

`create cluster` requires the following arguments:

- A `package_name`, supplied as the value of the `--package` option (short form: `-P`). This must be the name of a package previously registered using `add package`.
- A list (`process_host_list`) of MySQL Cluster processes, the hosts on which they are to run, and—optionally—their node IDs, supplied as the value of the `--processhosts` option (short form: `-R`), with list items separated by commas. As with other lists passed as option values in MySQL Cluster Manager commands, you must not use spaces before or after the commas.

Each item in the `process_host_list` consists of the name of a MySQL Cluster process—possibly suffixed with a colon (`:`) character followed by the process node ID—joined with the name of the host on which it is located using an amphora (`@`) sign (also sometimes known as the “at” sign). Permitted values for processes are `ndb_mgmd`, `ndbd`, and `mysqld`. When the cluster employs MySQL Cluster NDB 7.0 or later, you can also use `ndbmtid` as process name; in other words, a valid process name is the name of a MySQL Cluster process daemon binary. If node IDs are specified, they must be within the allowed range for the type of node defined.

To support running your own NDB API applications with a cluster under MySQL Cluster Manager, it is also possible to use `ndbapi` as a process type. Such applications can be connected to a managed cluster. Currently, MySQL Cluster Manager recognises only that an NDB API application is connected to the cluster; the NDB API application itself must be started, stopped, and configured manually.

It is also possible to specify one or more “free” `mysqld` and `ndbapi` processes without any hosts. To do this, simply use the wildcard `*` (asterisk character) in place of the hostname or IP address, as shown below:

- “Free” `mysqld` process: `mysqld@*`
- “Free” `ndbapi` process: `ndbapi@*`

It is also possible to specify a node ID for a “free” process. (If this is not specified, MySQL Cluster Manager assigns a suitable node ID automatically.)

A `mysqld` process or `ndbapi` process that is specified without a host in this fashion is permitted to connect to the cluster from any host that can access the cluster over the network. Otherwise, the process may connect to the cluster only from the specified host.

By convention, items in the `process_host_list` are listed according to the process type, in the following order:

1. Management node processes (`ndb_mgmd`)
2. Data node processes (*MySQL Cluster NDB 6.3*: `ndbd`; *MySQL Cluster NDB 7.0 and later*: `ndbd`, `ndbmtid`)
3. SQL node processes (`mysqld`)
4. Custom NDB API applications (`ndbapi`)

For information about writing your own NDB API applications, see [The NDB API](#), in the *MySQL Cluster API Developer Guide*.

While the order in which the items are listed does not affect whether the `create cluster` command succeeds, we suggest that you follow this convention for readability, as well as compatibility with other MySQL Cluster management tools such as `ndb_mgm`.

`create cluster` causes cluster node IDs to be assigned consecutively, in the order that the nodes are specified in the `process_host_list`, with node IDs for data node processes starting with 1, and node IDs for processes other than data node processes starting with 49. For MySQL Cluster Manager 1.3.3 and earlier, trying to manually assign node IDs less than 49 for `ndb_mgmd`, `mysqld`, or `ndbapi` fails with an error; the restriction, however, has been relaxed in two ways:

- For MySQL Cluster Manager 1.3.1 and later, the `--import` option causes this restriction to be ignored, as discussed later in this section.
- For MySQL Cluster Manager 1.3.4 and later, the restriction has now been totally lifted. However, you are still recommended to follow the best practice of reserving node ID 1 to 48 for data nodes.

Each host referenced in the list must be part of the site for which the package used in `create cluster` is defined.

For processes of types `mysqld` and `ndbapi`, the hostname is required, but not enforced in the running cluster. In other words, an `[api]` section is created in the cluster `config.ini` file, but no `HostName` parameter is specified; thus, the `mysqld` or `ndbapi` can connect from any host. (Currently, there is no way using MySQL Cluster Manager to specify that a `mysqld` or `ndbapi` process is restricted to connecting from a single host.)

- A name for the cluster. Once the cluster has been created, this name is used to refer to it in other cluster management commands such as `delete cluster`, `start cluster`, and `stop cluster`. Like other object names used with MySQL Cluster Manager, the `cluster_name` must be valid according to the rules given elsewhere in this document for identifiers (see [Chapter 4, MySQL Cluster Manager Client Commands](#)).

An additional `--verbose` option for this command causes `create cluster` to output extra information as it is executed, as shown later in this section.

The `--import` option, introduced in MySQL Cluster Manager 1.3.0, flags the cluster as being created as a target for importing a cluster created outside MySQL Cluster Manager. This option causes the cluster's status to appear as `import` in the output of `show status`, as shown here:

```
mcm> show status --process newcluster;
+-----+-----+-----+-----+-----+-----+
| NodeId | Process | Host | Status | Nodegroup | Package |
+-----+-----+-----+-----+-----+-----+
| 1      | ndb_mgmd | alpha | import |           | newpackage |
| 5      | ndbd     | beta  | import | n/a       | newpackage |
| 6      | ndbd     | gamma | import | n/a       | newpackage |
| 10     | mysqld   | delta | import |           | newpackage |
| 11     | ndbapi   | *     | import |           |             |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.04 sec)
```

Having the `import` status causes any of the commands `start cluster`, `restart cluster`, `start process`, and `stop process` to fail if they are executed before an `import cluster` command has been executed against this cluster. It is also not possible to execute `upgrade cluster` on a cluster having processes with `import` status. Other operations on this cluster continue to be performed normally.

Beginning with MySQL Cluster Manager 1.3.1, the `--import` option also overrides the usual requirement (effective until MySQL Cluster Manager 1.3.4) that node IDs assigned to processes other than data nodes must be greater than 48. (Bug #18181039)



Caution

While it is possible to import into a cluster that was created without this option, it is not advisable, since the cluster is not protected against accidentally performing any of the operations listed previously, which may result in confusing or misleading errors, and possibly other problems. For this reason, it is strongly recommended that you always use the `--import` option for creating the cluster in such cases.

For more information about importing clusters into MySQL Cluster Manager, including examples, see [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#).

Example

Consider the following command issued in the MySQL Cluster Manager client, which creates a cluster named `mycluster`:

```
mcm> create cluster --package=mypackage
-> --processhosts=ndb_mgmd@flundra,ndbd@tonfisk,ndbd@grindval,mysqld@flundra
-> mycluster;
+-----+
| Command result |
+-----+
| Cluster created successfully |
+-----+
1 row in set (7.71 sec)
```

As defined by the command just shown, `mycluster` consists of four nodes: a management node on host `flundra`; two data nodes—one on each of the hosts `tonfisk` and `grindval`; and one SQL node, also on host `flundra`.

Using the `--verbose` option causes the command to print output similar to that produced by the `list processes` command, as shown here:

```
mcm> create cluster --verbose --package=mypackage
-> --processhosts=ndb_mgmd@flundra,ndbd@tonfisk,ndbd@grindval,mysqld@flundra
-> mycluster;
+-----+
| NodeId | Name      | Host      |
+-----+
| 49     | ndb_mgmd | flundra   |
| 1      | ndbd     | tonfisk   |
| 2      | ndbd     | grindval  |
| 50     | mysqld   | flundra   |
+-----+
4 rows in set (0.32 sec)
```

You can also create this cluster in such a way that the `mysqld` process is permitted to connect to the cluster from any host able to reach the other cluster hosts over the network as shown here:

```
mcm> create cluster --package=mypackage
-> --processhosts=ndb_mgmd@flundra,ndbd@tonfisk,ndbd@grindval,mysqld@*
-> mycluster;
+-----+
| Command result |
+-----+
| Cluster created successfully |
+-----+
1 row in set (7.71 sec)
```

**Note**

In the case of a “free” `ndbapi` process, it is not necessary to have the MySQL Cluster Manager software installed on the host where the `ndbapi` process is running.

Configuration changes to the newly-created cluster can be made using the `set` command prior to starting the cluster. This is often preferable to doing after the cluster has been started, since `set` commands used to make configuration changes in a running cluster can require a rolling restart, and rolling restarts of clusters having many nodes or large quantities of data (or both) may take a great deal of time to complete.

**Note**

When creating a cluster having more than one `mysqld` process on the same host machine, MySQL Cluster Manager assigns the MySQL default port (3306) to each of them. Therefore, you must assign a unique port for each `mysqld` process in the cluster.

4.4.2 The `delete cluster` Command

```
delete cluster [--removedirs] cluster_name
```

This command deletes the cluster named `cluster_name`, removing it from the list of clusters managed by MySQL Cluster Manager.

`delete cluster` does *not* remove any MySQL Cluster binaries from hosts. However, it *does* remove the cluster configuration, data, and log files that reside in the MySQL Cluster Manager data repository.

This example demonstrates how to delete a cluster named `mycluster`:

```
mcm> delete cluster mycluster;
+-----+
| Command result          |
+-----+
| Cluster deleted successfully |
+-----+
1 row in set (1.22 sec)
```

A look at the MySQL Cluster Manager data repository (at `/opt/mcm_data/` in this case) shows that the folder that used to host the configuration, data, and log files for `mycluster` (`/opt/mcm_data/clusters/mycluster`) no longer exists:

```
shell> ls -l /opt/mcm_data/clusters
total 0
```

To remove the configuration and data files outside of the MySQL Cluster Manager data repository, `delete cluster` must be invoked with the `--removedirs` option, like this:

```
mcm> delete cluster --removedirs mycluster;
+-----+
| Command result          |
+-----+
| Cluster deleted successfully |
+-----+
1 row in set (1.22 sec)
```

For example, if one of the data node on `mycluster` has its data directory outside of the MySQL Cluster Manager data repository:

```
mcm> get Datadir mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value                                     | Process1 | NodeId1 | Process2 | NodeId2 | Level  | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
+-----+-----+-----+-----+-----+-----+
| DataDir | /home/dso/mycluster/cdata | ndbd      | 1        |          |          | Process |          |
| ...     |                             |           |          |          |          |         |          |
```

Deleting mycluster without using `--removedirs` does *not* remove the data directory for node 1:

```
shell> ls -l /home/dso/mycluster
total 4 drwxr-xr-x. 3 dso dso 4096 Sep 10 18:00 cdata
```

However, if the `--removedirs` option is used, the data directory for node 1 also gets removed:

```
shell> ls -l /home/dso/mycluster
total 0
```

`delete cluster` fails if the cluster to be deleted is running, as shown here:

```
mcm> delete cluster mycluster;
ERROR 5010 (OOMGR): All processes must be stopped to delete cluster mycluster
```

You must shut down the cluster first, using `stop cluster`.

4.4.3 The `list clusters` Command

```
list clusters site_name
```

This command lists all clusters defined for a given management site named `site_name`, together with the package used by each cluster. For example, the command shown here displays a list of all clusters defined for the site named `mysite`:

```
mcm> list clusters mysite;
+-----+-----+
| Cluster      | Package |
+-----+-----+
| mycluster    | m-7.1.26 |
| yourcluster  | y-7.1.26 |
| someothercluster | s-7.2.9 |
+-----+-----+
3 rows in set (2.07 sec)
```

If `site_name` is omitted, the command fails with an error, as shown here:

```
mcm> list clusters;
ERROR 6 (OOMGR): Illegal number of operands
```

4.4.4 The `list nextnodeids` Command

```
list nextnodeids cluster_name
```

MySQL Cluster Manager normally assigns IDs to new node processes automatically (although this can be overridden when issuing the `create cluster` or `add process` command). The `list nextnodeids` command can be used to see the next node ID that MySQL Cluster Manager has reserved for the next new process (of each possible process type) to be added to the cluster named `cluster_name`, which is required.

```
mcm> list nextnodeids mycluster;
+-----+-----+-----+-----+
| Category | NodeId Range | Next NodeId | Processes |
+-----+-----+-----+-----+
| Datanodes | 1 - 48       | 5           | ndbd, ndbmtid |
| Others    | 49 - 255    | 52          | ndb_mgmd, mysqld, ndbapi |
+-----+-----+-----+-----+
2 rows in set (0.07 sec)
```

4.4.5 The `restart cluster` Command

```
restart cluster cluster_name
```

This command performs a rolling restart (see [Performing a Rolling Restart of an NDB Cluster](#)) of the cluster named *cluster_name*. The cluster must already be running in order for this command to succeed. (For information about how to determine the operation state of the cluster, see [Section 4.4.6, “The `show status` Command”](#).)

For example, the command shown here performs a rolling restart of the cluster named `mycluster`:

```
mcm> restart cluster mycluster;
+-----+
| Command result |
+-----+
| Cluster restarted successfully |
+-----+
1 row in set (1 min 22.53 sec)
```

If the cluster is not already running, `restart cluster` fails with an error, as shown here:

```
mcm> show status --cluster mycluster;
+-----+-----+-----+
| Cluster | Status | Comment |
+-----+-----+-----+
| mycluster | stopped | |
+-----+-----+-----+
1 row in set (1.49 sec)

mcm> restart cluster mycluster;
ERROR 5009 (00MGR): Restart can not be performed as processes are
stopped in cluster mycluster
```



Note

Depending on the number of nodes and the amount of data stored in the cluster, a rolling restart can take a considerable amount of time, up to several hours for a cluster with a great many data nodes and a large amount of data. For this reason, you may want to execute this command with the `--background` option (short form `-B`) to allow it to run in the background, freeing the MySQL Cluster Manager client for other tasks.

Currently, there is no mechanism in MySQL Cluster Manager for performing system initial restarts of a cluster. This means that attributes that require an initial restart to be changed must be set before the cluster is started for the first time.

4.4.6 The `show status` Command

```
show status --cluster|-c cluster_name
show status --operation|-o cluster_name
show status --backup|-b cluster_name
show status --process|-r cluster_name
```

This command is used to check the status of clusters, cluster processes, backups, and commands issued in the MySQL Cluster Manager client. The type of status returned depends on which of the four options of `--cluster` (short form: `-c`), `--operation` (short form: `-o`), `--backup` (short form: `-b`), or `--process` (short form `-r`) is used with the command. (If none of these is used, `--cluster` is assumed.) These options are described in more detail in the next few paragraphs.

`--cluster`

When this option is used, `show status` reports on the status of the cluster named *cluster_name*, as shown in this example:

```
mcm> show status --cluster mycluster;
+-----+-----+-----+
| Cluster | Status | Comment |
+-----+-----+-----+
| mycluster | fully operational | |
+-----+-----+-----+
1 row in set (0.01 sec)
```

When used with the `--cluster` option (short form: `-c`), the output of this command consist of two columns. The `Cluster` column contains the name of the cluster. The `Status` column contains a description of the cluster's status; possible values and their meanings are shown in the following table:

Table 4.1 Status values shown by `show status --cluster`

Status Value	Meaning
<code>fully operational</code>	All cluster processes are running.
<code>operational</code>	All node groups are up and running, but at least one data node process (<code>ndbd</code> or <code>ndbmtbd</code>) is not running. The cluster is online, but you should determine why any “missing” data nodes are not running and correct the problem as soon as possible.
<code>non-operational</code>	The cluster is not operational, because at least one node group is offline. You must investigate and fix the problem or problems, then restart the cluster, before the cluster can be used for data storage and retrieval operations.
<code>stopped</code>	The cluster is not running, because it has been stopped by the user. This normally does not indicate any problem as such, but you must restart the cluster before it can be used by any applications.
<code>created</code>	The cluster has been created successfully using the <code>create cluster</code> command, but has never been started. You must start the cluster using the <code>start cluster</code> command before you can make use of it.
<code>unknown</code>	The MySQL Cluster Manager was unable to determine the cluster's status. This may or may not indicate a problem with the cluster; it is possible that the problem lies with one or more MySQL Cluster Manager agents or the MySQL Cluster Manager client. You should attempt to determine the status of the cluster by other means, such as using <code>show status --process</code> in the MySQL Cluster Manager client (described later in this section), or employing one of the commands available in the <code>ndb_mgm</code> client (see ndb_mgm — The NDB Cluster Management Client) such as <code>SHOW</code> or <code>ALL STATUS</code> .

`--operation` option

When the `--operation` option (short form: `-o`) is used, it causes `SHOW STATUS` to display the status of the latest command to be executed. This includes commands that were issued using the `--background` option (short form `-bg`). An example of this command is shown here:

```
mcm> show status --operation mycluster;
+-----+-----+-----+
```

```
| Command      | Status      | Description |
+-----+-----+-----+
| restart cluster | executing | <no message> |
+-----+-----+-----+
1 row in set (1.60 sec)
```

The output contains 3 columns, described in the following list:

- **Command.** The text of the command last issued (previous to the `show status --operation` command), less any options or arguments.
- **Status.** The current state of the command. Possible values and their meanings are listed later in this section.
- **Description.** In some cases, depending on the command and its status, this column may contain additional information. Otherwise, `<no message>` is displayed here.

Possible values for the `Status` column, together with descriptions of these values, are shown in the following table:

Table 4.2 Status values shown by `show status --operation`

Status Value	Description
<code>executing</code>	MySQL Cluster Manager is executing the command, but has not yet completed doing so.
<code>finished</code>	The command has executed (and completed) successfully.
<code>failed</code>	The command failed to execute. The <code>Description</code> column may contain information about the reason for the failure.
<code>unknown</code>	MySQL Cluster Manager was unable to determine the status of this command.

`--backup` option

When this option is used, `show status` reports on the status of the backup process for the cluster named `cluster_name`, as shown in the following examples:

```
mcm> show status --backup mycluster;
+-----+-----+
| Command result |
+-----+-----+
| No backup currently active in mycluster |
+-----+-----+
1 row in set (0.05 sec)
```

```
mcm> show status --backup mycluster;
+-----+-----+
| Command result |
+-----+-----+
| BackupId 5 currently active in mycluster |
+-----+-----+
1 row in set (0.09 sec)
```

`--process` option

When run with this option, `show status` returns information about each process in the cluster named `cluster_name`, as shown in this example:


```
mcm> show status --process mycluster;
+-----+-----+-----+-----+-----+
| Id    | Process | Host   | Status | Nodegroup |
+-----+-----+-----+-----+-----+
| 1     | ndb_mgmd | tonfisk | running |           |
| 2     | ndbd     | flundra | running | 0         |
| 3     | ndbd     | grindval | running | 0         |
| 4     | mysqld   | lax     | running |           |
+-----+-----+-----+-----+-----+
4 rows in set (1.67 sec)
```

When the `--process` option (short form: `-r`) is used with `show status`, the output contains 5 columns, described in the following list:

- **Id.** This is the node ID of the process as a node in cluster `cluster_name`.
- **Process.** The type of process, that is, the name of the corresponding MySQL Cluster executable. Allowed values are `ndb_mgmd`, `ndbd`, `ndbmtid`, and `mysqld`.
- **Host.** The hostname or IP address of the computer where the process is running.
- **Status.** The state or condition of this process. Possible values for this column are given later in this section.
- **Nodegroup.** If the `Process` is `ndbd` or `ndbmtid`—that is, if the process is a data node process—then this column shows the ID of the node group to which the process belongs. For any other value of `Process`, this column is empty.

Possible values for the `Status` column are shown in the following table, together with a description of what this value represents:

Table 4.3 Status values shown by `show status --process`

Status Value	Meaning
<code>running</code>	The process is running normally.
<code>stopped</code>	The process has been stopped by the user.
<code>added</code>	The process has been added to the cluster, but not yet started.
<code>starting</code>	The process has been started, but is not yet fully running. (For data nodes, you can determine which start phase the node is currently in by using the <code>status</code> command in the <code>ndb_mgm</code> client.)
<code>stopping</code>	The process has received a command to stop, and is now shutting down.
<code>failed</code>	The process has shut down unexpectedly (likely to have crashed). You should determine the cause for this unplanned shutdown, fix the problem, and restart the process as soon as possible.
<code>import</code>	The process is part of a cluster that was created for import, but the actual migration of processes and data from the original cluster has not yet taken place. <code>start process</code> and <code>stop process</code> commands fail for this process until this migration has occurred.

Status Value	Meaning
<code>unknown</code>	MySQL Cluster Manager is unable to establish the current status of this process. You should try to determine its status using other means.

You must supply the name of an existing cluster with this command, or else `show status` fails with an error, as shown here:

```
mcm> show status;
ERROR 6 (00MGR): Illegal number of operands

mcm> show status -c nosuchcluster;
ERROR 5001 (00MGR): Cluster nosuchcluster not defined
```



Important

Do not confuse this command with the MySQL `SHOW STATUS` statement, which has a different syntax, and which can be used only in the standard `mysql` client. The MySQL Cluster Manager client command accepts only those options shown at the beginning of this section, and does not accept a `LIKE` or `WHERE` clause.

4.4.7 The `start cluster` Command

```
start cluster [--initial|-i] cluster_name
```

This command starts the cluster named `cluster_name`, as shown in this example:

```
mcm> start cluster mycluster;
+-----+
| Command result          |
+-----+
| Cluster started successfully |
+-----+
1 row in set (45.37 sec)
```

In order for the command to succeed, the cluster named in the command must already exist; otherwise the command fails with the error `Cluster cluster_name not defined`, as shown here:

```
mcm> list sites;
+-----+-----+-----+-----+
| Site   | Port | Local | Hosts          |
+-----+-----+-----+-----+
| mysite | 1862 | Local | tonfisk,flundra,grindval,haj |
+-----+-----+-----+-----+
1 row in set (1.72 sec)

mcm> list clusters mysite;
+-----+-----+
| Cluster | Package |
+-----+-----+
| mycluster | mypackage |
+-----+-----+
1 row in set (1.70 sec)

mcm> start cluster yourcluster;
ERROR 5001 (00MGR): Cluster yourcluster not defined
```

In addition, the cluster must not already be running, as shown here:

```
mcm> show status --cluster mycluster;
+-----+-----+-----+
| Cluster | Status          | Comment |
+-----+-----+-----+
```

```
| mycluster | fully operational |          |
+-----+-----+-----+
1 row in set (0.01 sec)

mcm> start cluster mycluster;
ERROR 5005 (OOMGR): Cluster mycluster is running
```

A cluster created for import cannot be started until the import has been completed. See [Section 4.4.1, “The `create cluster` Command”](#), and [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#), for more information.

`--initial`

The `--initial` option (short form: `-i`) causes all cluster data node processes to be started as if `start process --initial` had been used, which means that all data nodes wipe their data and start with clean data node file systems. NDB tables that were previously stored in the cluster are lost.

Under normal circumstances, you should use this option to start a cluster only when either you do not wish to preserve any of its data (and want to make a clean start), or you intend to restore the cluster from backup to a known good state (see [Section 4.7.4, “The `restore cluster` Command”](#)). You should also be aware that no special warnings are printed by the `mcm` client when `--initial` is used with `start cluster`; the command is immediately executed.

For information about creating cluster backups, see [Section 4.7.2, “The `backup cluster` Command”](#). If you need to know which backups are available (if any), use `list backups`.

Using the `--initial` option with `start cluster` does not cause `mysql_install_db` to be executed, or the `mysql` system database to be altered or removed, on any SQL nodes. This option also does not remove MySQL Cluster Disk Data tablespace files; if their removal is needed, it must be performed manually.

4.4.8 The `stop cluster` Command

```
stop cluster cluster_name
```

This command stops the cluster named `cluster_name`, if it is running, as shown in this example:

```
mcm> stop cluster mycluster;
+-----+-----+-----+
| Command result |
+-----+-----+-----+
| Cluster stopped successfully |
+-----+-----+-----+
1 row in set (21.31 sec)
```

`stop cluster` fails if the cluster is not in an operational state (see [Section 4.4.6, “The `show status` Command”](#), for information about obtaining the cluster’s status):

```
mcm> show status --cluster mycluster;
+-----+-----+-----+
| Cluster | Status | Comment |
+-----+-----+-----+
| mycluster | stopped |
+-----+-----+-----+
1 row in set (0.01 sec)

mcm> stop cluster mycluster;
ERROR 5006 (OOMGR): Cluster mycluster is stopped
```

`stop cluster` cannot be used on a cluster created for import until the import has been completed. See [Section 4.4.1, “The `create cluster` Command”](#), and [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#), for more information.

4.4.9 The `upgrade cluster` Command

```
upgrade cluster [--package=|-P ]package_name
  [--set=attribute_assignment_list] cluster_name

attribute_assignment_list:
  attribute_assignment[,attribute_assignment][,...]

attribute_assignment:
  attribute_name:process_name[=value]
```

This command upgrades the cluster named `cluster_name` to the software package `package_name`. It accomplishes this by performing a rolling restart of the cluster.

The new package must be registered using `add package` before you can use it for an upgrade; otherwise, `upgrade cluster` fails with an error.

To use the command to perform an upgrade, the cluster must be in the `fully operational` status (you can check that using the command `show status --cluster cluster_name`). A cluster created for import cannot be upgraded until the import has been completed. See [Section 4.4.1, “The create cluster Command”](#), and [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#), for more information.

Suppose `mycluster` is using MySQL Cluster NDB 7.1.17, and the binaries are registered with a package named `7.1.17`, as shown by this `list clusters` command:

```
mcm> list clusters mysite;
+-----+-----+
| Cluster | Package |
+-----+-----+
| mycluster | 7.1.17 |
+-----+-----+
1 row in set (1.80 sec)
```

Now you wish to upgrade `mycluster` to MySQL Cluster NDB 7.1.19. Assuming that you have placed the NDB 7.1.19 binaries in the same directory on each host, the `add package` command to create a new package named `7.1.19` that contains these binaries might look something like this:

```
mcm> add package --basedir=/usr/local/ndb-7.1.19 7.1.19;
+-----+-----+
| Command result |
+-----+-----+
| Package added successfully |
+-----+-----+
1 row in set (0.88 sec)
```



Note

On Windows, you must replace any backslash (`\`) characters in the path used for the `add package` command's `--basedir` option with forward slashes (`/`). See [Section 4.3.1, “The add package Command”](#), for additional information and examples.

Both packages should now be listed in the output of the command `list packages mysite`. To perform the upgrade to the `7.1.19` package, use the `upgrade cluster` command as shown here:

```
mcm> upgrade cluster --package=7.1.19 mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Cluster upgraded successfully |
+-----+-----+
```

```
1 row in set (3 min 17.00 sec)
```

Once the `upgrade cluster` command has been successfully executed, you can verify that `mycluster` is now using the 7.1.19 package from the output of the appropriate `list clusters` command:

```
mcm> list clusters mysite;
+-----+-----+
| Cluster | Package |
+-----+-----+
| mycluster | 7.1.19 |
+-----+-----+
1 row in set (1.80 sec)
```

The command can perform major as well as minor series upgrades. Despite the name of this command, `upgrade cluster` can also be used to perform MySQL Cluster downgrades.

Not all upgrades and downgrades between different versions of MySQL Cluster are supported by the command. Three criteria must be met:

- The upgrade or downgrade must be supported by the MySQL Cluster versions involved. See the following sections in the MySQL Cluster manuals for lists of allowed upgrades and downgrades:
 - For MySQL Cluster 7.1: See the *MySQL 5.1 Reference Manual*.
 - For MySQL Cluster 7.2: See the *MySQL 5.5 Reference Manual*.
 - For MySQL Cluster 7.3 and 7.4: See [Upgrading and Downgrading NDB Cluster](#).
- Both the versions you upgrade or downgrade to and from must be supported by the version of MySQL Cluster Manager you are using (for example, upgrade from MySQL Cluster 6.3.8 to 7.3.8 must be performed manually, because MySQL Cluster Manager no longer supports MySQL Cluster 6.3).
- For MySQL Cluster 7.2.13 and before, the Cluster must NOT have [user privileges distributed across its SQL nodes](#), or the upgrade will have to be performed manually.

Starting from MySQL Cluster Manager 1.3.5, when using the `upgrade cluster` command, you can use the `--set` option to reconfigure your MySQL Cluster at the same time. This is particularly helpful when the upgrade requires configuration changes to your cluster. This option takes as its argument an attribute assignment list similar in format to that used with the `get` and `set` commands; see description of the `set` command on the proper way to formulate an attribute assignment list. For example: if you want to change the memory assigned to each data node for storing database records to 750M, specify that with the `--set` option in your `upgrade cluster` command:

```
mcm> upgrade cluster --package=7.1.19 --set=DataMemory:ndbd=750M mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Cluster upgraded successfully |
+-----+-----+
1 row in set (3 min 17.04 sec)
```



Note

Unlike the way you use the `set` command, an equal sign (=) immediately following the `--set` option is required.

4.5 MySQL Cluster Manager Configuration Commands

This section covers the commands used in the MySQL Cluster Manager for getting and setting values of various types used in MySQL Cluster configuration. We begin with a discussion of what we mean by the

term “configuration attribute”, and how this relates to the manual configuration of a MySQL Cluster using MySQL Cluster configuration parameters and MySQL Server options and variables that you may already be familiar with.

Configuration attributes.

Traditionally, when administering MySQL Cluster, it has been necessary to distinguish between 3 types of configuration data:

- *Configuration parameters* set in the MySQL Cluster global configuration file read by the management server (or servers), by convention named `config.ini`
- *Configuration variables* set in a running MySQL server (SQL node) by using the SQL `SET` statement in the `mysql` command-line client (or in another MySQL client application)
- *Configuration options* passed to MySQL Cluster executable programs when invoking them



Note

Configuration options passed to `mysqld` often have the effect of setting values for configuration variables, many—but not all—of which can be overridden in a running MySQL server using a MySQL client application such as `mysql`.

MySQL Cluster Manager simplifies this configuration scheme by treating all 3 types of configuration data as *attributes*, where the term “attribute” refers to a MySQL Cluster configuration parameter, a MySQL Server variable, or a command-line option used with one or more MySQL Cluster binary programs. It does this transparently, handling all necessary changes in a unified interface.

Suppose that you wish to know how much data memory is allocated to the data nodes in a given MySQL Cluster. Rather than having to determine that this is controlled using the `DataMemory` configuration parameter that is written in the `config.ini` file and then reading that file to find the value, you merely invoke the MySQL Cluster Manager `get` command, and MySQL Cluster Manager handles reading from the file for you, and displays the value without the necessity of opening the file in a separate application such as `more` or `less`. If you wish to change the amount of data memory allocated to the data nodes, you can issue a MySQL Cluster Manager `set` (or `reset`) command; MySQL Cluster Manager then writes the desired value to `config.ini`. If—as is the case with `DataMemory`—updating a configuration value in a running MySQL Cluster requires a rolling restart to be performed, MySQL Cluster Manager can perform this operation automatically so that the configuration change takes effect without further intervention required on the part of the operator.

Configuration attribute levels.

A configuration attribute value applies at one of the three levels, described here:

- *Default*: This value is always used by any MySQL Cluster process of the type or types (such as `ndbd` or `mysqld`) to which the attribute applies, unless this value is overridden by the user.
- *Process*: This value is used for all instances of a given type of MySQL Cluster process.
- *Instance*: This value is used for a specific instance of a MySQL Cluster process, the instance being identified by its MySQL Cluster node ID.

Default values are hard-coded into MySQL Cluster; you can override a default value for a given configuration attribute (using the `set` command) or reset a given attribute value to its default (using the `reset` command), but you cannot change a default value itself. You can set or reset an configuration attribute's value on either the process level or the instance level using a single `set` or `reset` command. Once you have set or reset the value of a configuration attribute, this value persists until it is changed by executing another `set` or `reset` command.

**Note**

When setting or resetting a configuration attribute value, you must specify the level at which the setting applies.

MySQL Cluster Manager determines what value to use for a configuration attribute relating to a given process by following these steps for each MySQL Cluster process:

(For each configuration attribute:)

1. Is an attribute value defined for the node ID of this process?
 - Yes: Use the value that was defined for this node ID, and exit.
 - No: Proceed to the next step.
2. Is an attribute value specified on the process level, that is, for all processes of this type?
 - Yes: Use the value that was specified for all processes of this type, and exit.
 - No: Use the default value that applies to processes of this type, and exit.

(In the steps just shown, “exit” can be taken to mean “If there are more configuration attributes applicable to this process that have not yet been set, proceed to the next attribute until there are no more attributes to be set for this process”.)

**Note**

The most recently specified value takes precedence. This means that if you set a configuration attribute for a specific process, then later specify a process-level value for this attribute, the process-level value is used for all processes of that type, including the instance for which you earlier set an instance-specific value.

Mandatory attributes.

Some attributes must be defined in the MySQL Cluster Manager at the process type or instance level for all processes of the applicable type or types for the cluster configuration to be valid. Such *mandatory attributes* may be changed, but not reset; in other words, the definition can be changed, but the definition itself cannot be removed entirely. Another way of stating this is that a mandatory attribute has no default value.

An example of a mandatory attribute is `NodeId`. If you try to reset a mandatory attribute, the attempt fails with an error, as shown here:

```
mcm> reset NodeId:ndb_mgmd:1 mycluster;
ERROR 6007 (00MGR): Config attribute NodeId is mandatory and cannot be reset
mcm> reset NodeId:ndbd:2 mycluster;
ERROR 6007 (00MGR): Config attribute NodeId is mandatory and cannot be reset
mcm> reset NodeId:mysqlid:4 mycluster;
ERROR 6007 (00MGR): Config attribute NodeId is mandatory and cannot be reset
```

Read-only attributes.

A *read-only attribute* is an attribute that must be defined by the MySQL Cluster Manager when a cluster is created. A read-only attribute can be neither changed nor reset by the user. This means that a read-only attribute is always a mandatory attribute.

One such attribute is `HostName`, which is read only for any type of MySQL Cluster process. Any attempt to change or reset a read-only attribute fails, as shown here:

```
mcm> reset HostName:ndb_mgmd mycluster;
```

```

ERROR 6008 (00MGR): Config attribute HostName is readonly and cannot be changed
mcm> reset HostName:ndbd mycluster;
ERROR 6008 (00MGR): Config attribute HostName is readonly and cannot be changed
mcm> reset HostName:mysqlld mycluster;
ERROR 6008 (00MGR): Config attribute HostName is readonly and cannot be changed

mcm> set HostName:ndb_mgmd mycluster;
ERROR 6008 (00MGR): Config attribute HostName is readonly and cannot be changed
mcm> set HostName:ndbd mycluster;
ERROR 6008 (00MGR): Config attribute HostName is readonly and cannot be changed
mcm> set HostName:mysqlld mycluster;
ERROR 6008 (00MGR): Config attribute HostName is readonly and cannot be changed

```

An attribute that is mandatory or read only is set when a cluster is created. Neither a mandatory attribute nor a read-only attribute can be reset. (Neither type of attribute has a default value other than what is set for it when the cluster is created.) A mandatory attribute can be changed at any time by the user; a read-only attribute cannot be changed once the cluster has been created. You can obtain a listing of mandatory and read-only attributes using the `get` command.

A listing of attribute properties also can be found in the output of `ndb_config --configinfo --xml` (see [ndb_config — Extract NDB Cluster Configuration Information](#)); for more complete information, see [Configuration of NDB Cluster](#).

MySQL Cluster Manager determines internally which attributes are considered read-only for reasons of cluster stability and performance. You can use the `get` command to see which attributes are read only.

4.5.1 The `get` Command

```

get [--include-defaults|-d] [filter_specification_list] cluster_name

filter_specification_list:
    filter_specification[,filter_specification][,...]

filter_specification:
    [attribute_name][:process_specification][+process_specification]

process_specification:
    [process_name][:process_id]

process_name:
    {ndb_mgmd|ndbd|ndbmt|mysqlld|ndbapi}

```

This command is used in the MySQL Cluster Manager client to obtain configuration attribute values from a MySQL Cluster. (See [Section 4.5, “MySQL Cluster Manager Configuration Commands”](#), for a definition of the term “attribute” as it applies in the MySQL Cluster Manager.) The output includes the following columns:

- **Name:** This column contains the name of the configuration attribute.
- **Value:** This column shows the attribute's current value.
- **Process1:** This column holds the process type to which the attribute applies. This is one of `ndb_mgmd`, `ndbd`, `ndbmt` (MySQL Cluster NDB 7.0 and later), or `mysqlld`.
- **Id1:** This is the process ID of the process to which the attribute applies.
- **Process2:** For attributes that require specifying two nodes, such as those relating to TCP/IP connections, this column shows the process type of the second node.
- **Id2:** For attributes that require specifying two nodes, this column shows the process ID for the second node.

- **Level:** This is the attribute process level. This value in this column can be `Default`, `Process`, or empty; if this column is empty, it means that the attribute applies on the instance level.
- **Comment:** This column is used to show whether the attribute is `Mandatory`, `Read only`, `Default attribute`, or user defined (in which case the `Comment` column is empty).

By default, `get` returns only those attributes that have been set explicitly, either by the MySQL Cluster Manager itself, or by the user. In other words, it shows only attributes that are mandatory (including read-only attributes), or that have been set by the user after the cluster was created. Hereafter in this discussion, we refer to these as “non-default attributes”.

Thus, prior to setting any configuration attributes, you can obtain a list of all mandatory and read-only attributes by running the simplest possible form of this command, as shown here:

```
mcm> get mycluster;
mcm> get mycluster\G
***** 1. row *****
      Name: DataDir
      Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/1/data
Process1: ndbmysd
      NodeId1: 1
Process2:
      NodeId2:
      Level:
      Comment:
***** 2. row *****
      Name: HostName
      Value: torsk
Process1: ndbmysd
      NodeId1: 1
Process2:
      NodeId2:
      Level:
      Comment: Read only
***** 3. row *****
      Name: NodeId
      Value: 1
Process1: ndbmysd
      NodeId1: 1
Process2:
      NodeId2:
      Level:
      Comment: Read only
***** 4. row *****
      Name: DataDir
      Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/2/data
Process1: ndbmysd
      NodeId1: 2
Process2:
      NodeId2:
      Level:
      Comment:
***** 5. row *****
      Name: HostName
      Value: torsk
Process1: ndbmysd
      NodeId1: 2
Process2:
      NodeId2:
      Level:
      Comment: Read only
***** 6. row *****
      Name: NodeId
      Value: 2
Process1: ndbmysd
```

The `get` Command

```
NodeId1: 2
Process2:
NodeId2:
Level:
Comment: Read only
***** 7. row *****
Name: DataDir
Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/49/data
Process1: ndb_mgmd
NodeId1: 49
Process2:
NodeId2:
Level:
Comment:
***** 8. row *****
Name: HostName
Value: torsk
Process1: ndb_mgmd
NodeId1: 49
Process2:
NodeId2:
Level:
Comment: Read only
***** 9. row *****
Name: NodeId
Value: 49
Process1: ndb_mgmd
NodeId1: 49
Process2:
NodeId2:
Level:
Comment: Read only
...
***** 24. row *****
Name: ndbcluster
Value:
Process1: mysqld
NodeId1: 51
Process2:
NodeId2:
Level:
Comment: Read only
***** 25. row *****
Name: NodeId
Value: 51
Process1: mysqld
NodeId1: 51
Process2:
NodeId2:
Level:
Comment: Read only
***** 26. row *****
Name: port
Value: 3307
Process1: mysqld
NodeId1: 51
Process2:
NodeId2:
Level:
Comment:
***** 27. row *****
Name: socket
Value: /tmp/mysql.mycluster.51.sock
Process1: mysqld
NodeId1: 51
```

```

Process2:
  NodeId2:
  Level:
  Comment:
***** 28. row *****
  Name: tmpdir
  Value: /home/jon/bin/mcm/mcm_data/clusters/mycluster/51/data/tmp
Process1: mysqld
  NodeId1: 51
Process2:
  NodeId2:
  Level:
  Comment:
***** 29. row *****
  Name: NodeId
  Value: 52
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level:
  Comment: Read only
29 rows in set (0.05 sec)

```

On Windows, no substitutions for backslashes or other characters used in values of paths reported by the `get` command is performed. However, it is possible to see forward slashes used in such paths if the values were set using the `set` command. See [Setting Attributes Containing Paths on Windows \[125\]](#), for more information.

Although a `socket` attribute is shown for `mysqld` nodes in the `get` output from the previous example and is not marked `Read only`, MySQL Cluster Manager does not support socket files on Windows. For this reason; you should not attempt to set `socket` attributes for Windows `mysqld` processes using MySQL Cluster Manager.

To include default values for attributes that have not (or not yet) been set explicitly, you can invoke this command with the `--include-defaults` option (short form: `-d`), as shown here (in part):

```

mcm> get --include-defaults mycluster\G
***** 1. row *****
  Name: __ndbmt_classic
  Value: NULL
Process1: ndbmt
  NodeId1: 1
Process2:
  NodeId2:
  Level: Default
  Comment:
***** 2. row *****
  Name: __ndbmt_lqh_threads
  Value: NULL
Process1: ndbmt
  NodeId1: 1
Process2:
  NodeId2:
  Level: Default
  Comment:
***** 3. row *****
  Name: __ndbmt_lqh_workers
  Value: NULL
Process1: ndbmt
  NodeId1: 1
Process2:
  NodeId2:
  Level: Default

```

The `get` Command

```
Comment:
***** 4. row *****
  Name: Arbitration
  Value: Default
Process1: ndbmt
  NodeId1: 1
Process2:
  NodeId2:
  Level: Default
Comment:
***** 5. row *****
  Name: ArbitrationTimeout
  Value: 7500
Process1: ndbmt
  NodeId1: 1
Process2:
  NodeId2:
  Level: Default
Comment:
...

***** 1094. row *****
  Name: DefaultOperationRedoProblemAction
  Value: queue
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level: Default
Comment:
***** 1095. row *****
  Name: ExecuteOnComputer
  Value: NULL
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level: Default
Comment:
***** 1096. row *****
  Name: HeartbeatThreadPriority
  Value: NULL
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level: Default
Comment:
***** 1097. row *****
  Name: HostName
  Value:
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level: Default
Comment:
***** 1098. row *****
  Name: MaxScanBatchSize
  Value: 262144
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level: Default
Comment:
```

The `get` Command

```
***** 1099. row *****
  Name: NodeId
  Value: 52
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level:
  Comment: Read only
***** 1100. row *****
  Name: TotalSendBufferMemory
  Value: 0
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level: Default
  Comment:
***** 1101. row *****
  Name: wan
  Value: false
Process1: ndbapi
  NodeId1: 52
Process2:
  NodeId2:
  Level: Default
  Comment:
1101 rows in set (0.09 sec)
```

As you can see, the output from this `get` command is quite long (and the number of rows generated increases with the number of nodes in the cluster.) However, it is possible to filter the output so that you can view only the attribute or attributes in which you are interested. This can be done by using a comma-separated list of one or more filter specifications. A filter specification is defined as shown here (condensed from that given at the beginning of this section, but effectively the same):

```
[attribute_name][:[process_name][:process_id]]
```

Filtering can be applied per attribute, per process type, and per process instance. We now provide some examples illustrating the use of such filters.

To obtain the value of a given attribute for all processes to which it applies in the cluster, you need only use the name of the attribute as a filter. For example, to obtain the `HostName` of all processes in the cluster named `mycluster`, you can execute the command shown here:

```
mcm> get HostName mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name   | Value   | Process1 | NodeId1 | Process2 | NodeId2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| HostName | flundra | ndbd     | 1       |           |         |      | Read only |
| HostName | tonfisk | ndbd     | 2       |           |         |      | Read only |
| HostName | grindval | ndb_mgmd | 49      |           |         |      | Read only |
| HostName | haj     | mysql    | 50      |           |         |      | Read only |
| HostName | torsk   | mysql    | 51      |           |         |      | Read only |
+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.04 sec)
```

To obtain the value of a given attribute for all processes of a given type, you can specify a filter of the form `attribute_name:process_name`. The following command retrieves the `HostName` of all `ndbd` processes (only) in the cluster `mycluster`:

```
mcm> get HostName:ndbd mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name   | Value   | Process1 | Id1  | Process2 | Id2  | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

The `get` Command

```
| HostName | flundra | ndbd | 1 | | | | Readonly |
| HostName | tonfisk | ndbd | 2 | | | | Readonly |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.12 sec)
```

To retrieve the value of a given attribute for a particular instance of a process, you can use a filter that takes the form `attribute_name:process_name:process_id`. For example, you can use the following command to obtain the hostname for the process having `2` as its process ID:

```
mcm> get HostName:ndbd:2 mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value  | Process1 | Id1  | Process2 | Id2  | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| HostName  | tonfisk | ndbd     | 2    |          |      |      | Readonly |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (1.67 sec)
```

The command works the same if the process type is omitted:

```
mcm> get HostName::2 mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value  | Process1 | Id1  | Process2 | Id2  | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| HostName  | tonfisk | ndbd     | 2    |          |      |      | Readonly |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (1.67 sec)
```

You can obtain information about multiple attributes within a single `get` command by specifying a list of filters, separated by commas. *Each filter in the list must be a complete, valid filter.* The command shown here retrieves the `HostName` and `DataDir` for all processes in `mycluster`:

```
mcm> get HostName,DataDir mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value  | Process1 | NodeId1 | Process2 | NodeId2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| DataDir   | /opt/c1data | ndbd     | 1      |          |          |      |          |
| HostName  | flundra   | ndbd     | 1      |          |          |      | Read only |
| DataDir   | /opt/c2data | ndbd     | 2      |          |          |      |          |
| HostName  | tonfisk   | ndbd     | 2      |          |          |      | Read only |
| DataDir   | /opt/c49data | ndb_mgmd | 49     |          |          |      |          |
| HostName  | grindval  | ndb_mgmd | 49     |          |          |      | Read only |
| datadir   | /opt/c50data | mysqld   | 50     |          |          |      |          |
| HostName  | haj       | mysqld   | 50     |          |          |      | Read only |
| datadir   | /opt/c51data | mysqld   | 51     |          |          |      |          |
| HostName  | torsk     | mysqld   | 51     |          |          |      | Read only |
+-----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.05 sec)
```

To retrieve the values of `HostName` and `DataDir` for only the data nodes in `mycluster`, you can use the `get` command shown here:

```
mcm> get HostName:ndbd,DataDir:ndbd mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value  | Process1 | Id1  | Process2 | Id2  | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| DataDir   | /opt/c2data | ndbd     | 1    |          |      |      | Read only |
| HostName  | tonfisk     | ndbd     | 1    |          |      |      | Read only |
| DataDir   | /opt/c3data | ndbd     | 2    |          |      |      | Read only |
| HostName  | flundra     | ndbd     | 2    |          |      |      | Read only |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (1.36 sec)
```

In the example just shown, each filter includes a process type specifier. If you omit this specifier from one of the filters, you obtain a result that you might not expect:

The `get` Command

```
mcm> get HostName,DataDir:ndbd mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
HostName	grindval	ndb_mgmd	49				Read only
DataDir	/opt/c2data	ndbd	1				
HostName	tonfisk	ndbd	1				Read only
DataDir	/opt/c3data	ndbd	2				
HostName	flundra	ndbd	2				Read only
HostName	haj	mysqld	50				Read only
HostName	torsk	mysqld	51				Read only

```
6 rows in set (0.58 sec)
```

The filter list `HostName,DataDir:ndbd` is perfectly valid. However, it actually consists of the filters `HostName` and `DataDir:ndbd`—in other words, it means “the `HostName` for all processes, and the `DataDir` for `ndbd` processes”.

Suppose you wish to obtain the values for `HostName` for just the `ndb_mgmd` and `mysqld` processes in `mycluster`. You might be tempted to try using something like `HostName:ndb_mgmd,mysqld` for the filter list, but this does not work, as you can see here:

```
mcm> get HostName:ndb_mgmd,mysqld mycluster;
```

ERROR 6003 (00MGR): No such config variable mysqld for process

This is due to the fact that each filter in the filter list must be a valid filter, and must include an attribute name. (In the filter list just shown, MySQL Cluster Manager tries to interpret the first string following the comma as an attribute name.) The correct filter list to use in a `get` command for retrieving the `HostName` for the `ndb_mgmd` and `mysqld` processes in `mycluster` is shown in this example:

```
mcm> get HostName:ndb_mgmd,HostName:mysqld mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
HostName	grindval	ndb_mgmd	49				Read only
HostName	haj	mysqld	50				Read only
HostName	torsk	mysqld	51				Read only

```
2 rows in set (0.21 sec)
```

It is also possible to obtain a list of attributes and their values for a given process type or instance of a process. For a given process type, use a filter having the form `:process_name`. For example, to retrieve all non-default attributes applying to `ndbd` processes in a cluster named `mycluster`, you can use the filter `:ndbd`, as shown here:

```
mcm> get :ndbd mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
DataDir	/opt/c2data	ndbd	1				
HostName	tonfisk	ndbd	1				Read only
NodeId	1	ndbd	1				Read only
DataDir	/opt/c3data	ndbd	2				
HostName	flundra	ndbd	2				Read only
NodeId	2	ndbd	2				Read only

```
6 rows in set (0.77 sec)
```

(The example just shown assumes that no attributes are set to non-default values.)

To get a list of all non-default attributes for a single instance of a process, use a filter having the form `:process_name:process_id`, as shown in this example, which retrieves all non-default attributes for the `ndbd` process having `2` as its process ID:

The `get` Command

```
mcm> get :ndbd:2 mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
DataDir	/opt/c2data	ndbd	2				
HostName	flundra	ndbd	2				Read only
NodeId	2	ndbd	2				Read only

```
4 rows in set (0.32 sec)
```

If you try to obtain values for an attribute which you know is supported by your MySQL Cluster version, but the result is empty, this almost certainly means that it is a default attribute which either has not been changed since the cluster was created or has been reset. In order to view default attributes using `get`, you must execute the command using the `--include-defaults` option (short form: `-d`).

Suppose you want to see how much `DataMemory` is configured for the `ndbd` processes in the cluster named `mycluster`, and you execute what appears to be the correct `get` command, but an empty result is returned, as shown here:

```
mcm> get DataMemory:ndbd mycluster;
```

```
Empty set (1.19 sec)
```

This means that the `DataMemory` attribute has its default value for all data nodes in the cluster. If you do not recall what this value is, you can determine it easily by repeating the same command with the addition of the `--include-defaults` (`-d`) option:

```
mcm> get --include-defaults DataMemory:ndbd mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
DataMemory	83886080	ndbd	1			Default	
DataMemory	83886080	ndbd	2			Default	

```
2 rows in set (0.62 sec)
```

Now suppose that you increase the `DataMemory` to 500 megabytes per data node, then repeat the `get` command to verify the new value:

```
mcm> set DataMemory:ndbd=500M mycluster;
```

```
-----+
| Command result          |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.77 sec)
```

```
mcm> get --include-defaults DataMemory:ndbd mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
DataMemory	500M	ndbd	1			Process	
DataMemory	500M	ndbd	2			Process	

```
2 rows in set (1.46 sec)
```

You can see that, not only has the `Value` column in the `get` command output been updated to the new value, but the `Level` column has also been updated from `Default` to `Process`. This means that you no longer need the `--include-defaults` option to view this attribute, as shown here:

```
mcm> get DataMemory:ndbd mycluster;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
------	-------	----------	-----	----------	-----	-------	---------


```

+-----+-----+-----+-----+-----+-----+-----+-----+
| DataMemory | 500M | ndbd | 1 | | | Process | |
| DataMemory | 500M | ndbd | 2 | | | Process | |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.63 sec)

```

However, if you reset `DataMemory` (also on the process level), this is no longer the case. Then, `DataMemory` once again assumes its default value, after which you must use the `--include-defaults` option to retrieve it, as shown in this example:

```

mcm> reset DataMemory:ndbd mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Command result |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Cluster reconfigured successfully |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (7.65 sec)

mcm> get DataMemory:ndbd mycluster;
Empty set (1.76 sec)

mcm> get --include-defaults DataMemory:ndbd mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value      | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| DataMemory    | 83886080   | ndbd     | 1   |          |     | Default |         |
| DataMemory    | 83886080   | ndbd     | 2   |          |     | Default |         |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (1.01 sec)

```

For more information about these commands, see [Section 4.5.3, “The set Command”](#), and [Section 4.5.2, “The reset Command”](#).

Starting from MySQL Cluster Manager 1.3.4, the `get` command also tags multi-entry replication attributes as so in the `Comment` column; for example:

```

mcm> get replicate_ignore_table:mysqlqd mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value      | Process1 | NodeId1 | Process2 | NodeId2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| replicate_ignore_table | mydb.t1    | mysqlqd  | 50      |          |         |       | Multi-entry |
| replicate_ignore_table | mydb.t50   | mysqlqd  | 50      |          |         |       | Multi-entry |
| replicate_ignore_table | mydb.mytable | mysqlqd  | 50      |          |         | Process | Multi-entry |
| replicate_ignore_table | mydb.t51   | mysqlqd  | 51      |          |         |       | Multi-entry |
| replicate_ignore_table | mydb.mytable | mysqlqd  | 51      |          |         | Process | Multi-entry |
+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.05 sec)

```

On how to reset multi-entry attributes, see [Section 4.5.2, “The reset Command”](#).

The `get` command does not normally display configuration attributes applying to TCP or SHM connections. However, such attributes can be set in the MySQL Cluster Manager client (using the `set` command); and once they have been set, they are displayed by applicable `get` commands. See [Setting TCP Connection Attributes](#), which provides an example of this.

4.5.2 The `reset` Command

```

reset filter_specification_list cluster_name

filter_specification_list:
    filter_specification[,filter_specification][,...]

filter_specification:
    attribute_name[:process_specification][+process_specification]

```

```
process_specification:
  [process_name][:process_id]

process_name:
  {ndb_mgmd|ndbd|ndbmtid|mysqld|ndbapi}
```

This command resets an attribute to its default value. Attributes can be set on either the process level or instance level. To reset an attribute on the process level, use a filter specification having the form `attribute_name:process_name`, where `attribute_name` is the name of the attribute to be reset, and `process_name` is the name of a MySQL Cluster process. To reset a configuration attribute on the instance level, use a filter specification of the form `attribute_name:process_name:process_id`, where `process_id` is the process ID.

You cannot issue a `reset` command that resets all values for a given configuration attribute regardless of process type; each `reset` command must specify a process type or instance of a process. Otherwise, the command fails, as shown here:

```
mcm> reset DataMemory mycluster;
ERROR 3 (00MGR): Illegal syntax
```

You also cannot revert all configuration attributes for a given process type or instance of a process using a single filter specification; you must always include the name of the attribute to be reset. Otherwise, the `reset` command fails, as shown here:

```
mcm> reset :ndbd mycluster;
ERROR 3 (00MGR): Illegal syntax

mcm> reset :ndbd:3 mycluster;
ERROR 3 (00MGR): Illegal syntax
```

Suppose that the data memory for all `ndbd` processes in the cluster named `mycluster` has been set to 500 MB, as shown in the output of this `get` command:

```
mcm> get DataMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value | Process1 | Id1 | Process2 | Id2 | Level   | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| DataMemory | 500M  | ndbd     | 2   |           |     | Process |         |
| DataMemory | 500M  | ndbd     | 3   |           |     | Process |         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (1.91 sec)
```

We can see from the entries in the `Level` column that the `DataMemory` setting for both `ndbd` processes applies on the process level. A process-level setting cannot be reset on the instance level, as shown here:

```
mcm> reset DataMemory:ndbd:2 mycluster;
ERROR 6010 (00MGR): No matching user defined setting was
found for config attribute DataMemory
mcm> reset DataMemory:ndbd:3 mycluster;
ERROR 6010 (00MGR): No matching user defined setting was
found for config attribute DataMemory
```

The following `reset` command also does not work, although you might think that it would do so, since it attempts to reset the attribute's value for both `ndbd` processes:

```
mcm> reset DataMemory:ndbd:2,DataMemory:ndbd:3 mycluster;
ERROR 6010 (00MGR): No matching user defined setting was
found for config attribute DataMemory
```

The previous command fails because MySQL Cluster Manager regards this as an attempt to apply two instance-level configuration changes. Because the `DataMemory` setting is a process-level setting, you

must instead reset `DataMemory` to its default value on the process level; you can do this by using the filter specification `DataMemory:ndbd` in the `reset` command, as shown here:

```
mcm> reset DataMemory:ndbd mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (6.16 sec)
```

If you execute the same `get` command as shown previously, the result is now empty:

```
mcm> get DataMemory mycluster;
Empty set (0.74 sec)
```

This is because the `get` command by default does not report default values. To retrieve the `DataMemory` values after resetting them, you must invoke `get` using the `--include-defaults` (short form: `-d`) option:

```
mcm> get --include-defaults DataMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value   | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| DataMemory | 83886080 | ndbd     | 2   |           |     | Default |         |
| DataMemory | 83886080 | ndbd     | 3   |           |     | Default |         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (1.21 sec)
```

The `DataMemory` values are now included in the output, and are marked with the word `Default` in the `Comments` column.

Now suppose that the `mysqld` configuration attribute `wait_timeout` for the `mysqld` process having the ID 4 in the cluster named `mycluster` has previously been set to the value 200 as shown here, and that no other changes have been to this attribute:

```
mcm> set wait_timeout:mysql:4=200 mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.78 sec)

mcm> get -d wait_timeout:mysql:4 mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| wait_timeout | 200 | mysql | 4 |           |     |       |         |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.98 sec)
```

Because the `Level` column is empty, we know that this setting applies on the instance level. If you try to reset it on the process level, the attempt fails, as shown here:

```
mcm> reset wait_timeout:mysql mycluster2;
ERROR 6010 (00MGR): No matching user defined setting was
found for config attribute wait_timeout
```

If you wish to reset this attribute to its default value, you must use the `reset` command with the instance-level filter specification `wait_timeout:mysql:4`, as shown here:

```
mcm> reset wait_timeout:mysql:4 mycluster;
```

The `reset` Command

```
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.61 sec)
```

Once you have reset `wait_timeout`, it no longer appears in the output of the earlier `get` command:

```
mcm> get wait_timeout:mysql mycluster;
Empty set (1.42 sec)
```

This is because the default behavior of the `get` command is to display only those values that have been set either by the MySQL Cluster Manager or by the user. Since `wait_timeout` has been allowed to revert to its default value, you must use the `--include-defaults` (short form: `-d`) option to retrieve it, as shown here:

```
mcm> get -d wait_timeout:mysql mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level   | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| wait_timeout | 28800 | mysql    | 4   |           |     | Default |         |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (1.66 sec)
```

Now consider a situation in which process-level and instance-level settings have been made to a configuration attribute; in this example, we use `IndexMemory`. First, verify that `IndexMemory` is set to its default value for all data node processes (in this case, there are two of them):

```
mcm> get -d IndexMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level   | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| IndexMemory  | 18874368 | ndbd    | 2   |           |     | Default |         |
| IndexMemory  | 18874368 | ndbd    | 3   |           |     | Default |         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (1.24 sec)
```

Now apply both a process-level change and an instance-level change to this attribute. You can do this with a single `set` command, as shown here:

```
mcm> set IndexMemory:ndbd=500M,IndexMemory:ndbd:3=750M mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.29 sec)
```

Because the process-level change was specified first, it is overridden for the `ndbd` process by the instance-level change specified second. The output from the following `get` command confirms that this is the case:

```
mcm> get IndexMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level   | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| IndexMemory  | 500M  | ndbd    | 2   |           |     | Process |         |
| IndexMemory  | 750M  | ndbd    | 3   |           |     |         |         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.85 sec)
```

If the instance-level `IndexMemory` setting for the `ndbd` process with process ID 3 is reset, the process-level setting still applies, as shown here:

```
mcm> reset IndexMemory:ndbd:3 mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (6.41 sec)

mcm> get IndexMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| IndexMemory   | 500M  | ndbd     | 2  |           |     | Process |         |
| IndexMemory   | 500M  | ndbd     | 3  |           |     | Process |         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (1.09 sec)
```

Now, re-apply the instance-level `IndexMemory` setting, and verify using `get` that it has taken effect:

```
mcm> set IndexMemory:ndbd:3=750M mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (6.79 sec)

mcm> get IndexMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| IndexMemory   | 500M  | ndbd     | 2  |           |     | Process |         |
| IndexMemory   | 750M  | ndbd     | 3  |           |     |         |         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (1.76 sec)
```

If you reset the process-level setting, the instance-level setting remains, and only the `ndbd` process having process ID 2 has its `IndexMemory` reset to the default value; the instance-level setting remains in effect, as you can see from the following sequence of commands:

```
mcm> reset IndexMemory:ndbd mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.36 sec)

mcm> get -d IndexMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| IndexMemory   | 18874368 | ndbd     | 2  |           |     | Default |         |
| IndexMemory   | 750M    | ndbd     | 3  |           |     |         |         |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.10 sec)
```



Note

If the order of the specifiers in the original command that set `IndexMemory` had been reversed as `IndexMemory:ndbd:3=750M, IndexMemory:ndbd=500M`, the instance-level change would have been overridden by the process-level change, and the resulting `IndexMemory` setting for both `ndbd` processes would be `500M`. As discussed elsewhere, a process-level setting made after an instance-

level setting that affects the same process completely removes the instance-level setting; the instance-level setting is not preserved, and resetting the attribute on the process level merely restores the default setting for all processes of that type. See [Section 4.5, “MySQL Cluster Manager Configuration Commands”](#), for more information.

Starting from MySQL Cluster Manager 1.3.4, the `get` and `reset` commands fully support multi-entry replication attributes; for example, if the `replicate_ignore_table` attribute has multiple entries:

```
mcm> get replicate_ignore_table:mysql mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value    | Process1 | NodeId1 | Process2 | NodeId2 | Level  | Comment  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| replicate_ignore_table | mydb.t1  | mysql    | 50      |           |          |        | Multi-entry |
| replicate_ignore_table | mydb.t50 | mysql    | 50      |           |          |        | Multi-entry |
| replicate_ignore_table | mydb.mytable | mysql    | 50      |           |          | Process | Multi-entry |
| replicate_ignore_table | mydb.t51 | mysql    | 51      |           |          |        | Multi-entry |
| replicate_ignore_table | mydb.mytable | mysql    | 51      |           |          | Process | Multi-entry |
+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.05 sec)
```

Without specifying a node ID, all the attribute's entries associated with the specified process type are reset with the following command:

```
mcm> reset replicate_ignore_table:mysql mycluster; # removes all process level entries
+-----+-----+
| Command result |
+-----+-----+
| Cluster reconfigured successfully |
+-----+-----+
1 row in set (0.47 sec)

mcm> get replicate_ignore_table:mysql mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value    | Process1 | NodeId1 | Process2 | NodeId2 | Level  | Comment  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| replicate_ignore_table | mydb.t1  | mysql    | 50      |           |          |        | Multi-entry |
| replicate_ignore_table | mydb.t50 | mysql    | 50      |           |          |        | Multi-entry |
| replicate_ignore_table | mydb.t51 | mysql    | 51      |           |          |        | Multi-entry |
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.08 sec)
```

With a node ID specified, only the instance entries associated with the node ID are reset by the following command:

```
mcm> reset replicate_ignore_table:mysql:51 mycluster; # removes all instance level entries for nodeid 51
+-----+-----+
| Command result |
+-----+-----+
| Cluster reconfigured successfully |
+-----+-----+
1 row in set (0.57 sec)

mcm> get replicate_ignore_table:mysql mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value    | Process1 | NodeId1 | Process2 | NodeId2 | Level  | Comment  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| replicate_ignore_table | mydb.t1  | mysql    | 50      |           |          |        | Multi-entry |
| replicate_ignore_table | mydb.t50 | mysql    | 50      |           |          |        | Multi-entry |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.09 sec)
```

Resetting TCP Connection Attributes. Certain configuration attributes, such as those relating to TCP connections, apply to connections between processes rather than to individual processes or individual

process types. As shown elsewhere (see [Setting TCP Connection Attributes](#)), when you set such an attribute on the process level using MySQL Cluster Manager, this means that the attribute applies to all connections between the two types of processes specified when issuing the `set` command. It is also possible to set such an attribute on the instance level, in which case it applies only to a single connection between two process instances.

Similarly, it is possible to reset such an attribute on either the process or instance level, depending on the level or levels at which it was set. In either case, an extended form of the process specifier is required, just as it is when setting an attribute that applies to a connection between processes. Assume that the `SendBufferMemory` attribute has previously been set for all connections between the two `ndbd` processes and the two `mysqld` processes that are found in a MySQL Cluster named `mycluster2`, as shown in the output of this `get` command:

```
mcm> get SendBufferMemory mycluster2;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
SendBufferMemory	4M	ndbd	2	mysqld	4	Process	
SendBufferMemory	4M	ndbd	2	mysqld	5	Process	
SendBufferMemory	4M	ndbd	3	mysqld	4	Process	
SendBufferMemory	8M	ndbd	3	mysqld	5		

```
4 rows in set (0.59 sec)
```

Suppose that you wish to reset `SendBufferMemory` only for the connection between the `ndbd` process having process ID 3 and the `mysqld` process having process ID 5. The `SendBufferMemory` setting that applies to this connection is specified on the instance level, as you can see because the `Level` column value corresponding to this connection is empty; this means that it is possible to reset this value on the instance level. You can do this using the `reset` command shown here:

```
mcm> reset SendBufferMemory:ndbd:3+mysqld:5 mycluster2;
```

```
Command result
```

```
Cluster reconfigured successfully
```

```
1 row in set (7.03 sec)
```

You can verify that the attribute was reset using the `get` command. However, as noted previously, once the instance-level setting has been removed, the process-level setting for this attribute again takes effect, so that the same setting applies to all connections between `ndbd` and `mysqld` processes, as shown here:

```
mcm> get SendBufferMemory mycluster2;
```

Name	Value	Process1	Id1	Process2	Id2	Level	Comment
SendBufferMemory	4M	ndbd	2	mysqld	4	Process	
SendBufferMemory	4M	ndbd	2	mysqld	5	Process	
SendBufferMemory	4M	ndbd	3	mysqld	4	Process	
SendBufferMemory	4M	ndbd	3	mysqld	5	Process	

```
4 rows in set (0.87 sec)
```

To reset this attribute on the process level, you can use the following `reset` command:

```
mcm> reset SendBufferMemory:ndbd+mysqld mycluster2;
```

```
Command result
```

```
Cluster reconfigured successfully
```

```
1 row in set (8.01 sec)
```

You can verify that the attribute has been reset for all connection between `ndbd` processes and `mysqld` processes, by using the `get` command, as shown here:

```
mcm> get -d SendBufferMemory mycluster2;
Empty set (1.39 sec)
```

As noted elsewhere in this manual (see [Section 4.5.1, “The `get` Command”](#)), the empty result set is to be expected in this case, even when `get` is invoked using the `--include-defaults` (or `-d`) option, because the MySQL Cluster Manager client does not display attributes that appear in the `[tcp]` or `[shm]` sections of the `config.ini` configuration file if they have not been explicitly set by the user.

4.5.3 The `set` Command

```
set attribute_assignment_list cluster_name

attribute_assignment_list:
    attribute_assignment[,attribute_assignment][,...]

attribute_assignment:
    attribute_name:process_specification[+process_specification][=value]

process_specification:
    [process_name][:process_id]

process_name:
    {ndb_mgmd|ndbd|ndbmtid|mysqld|ndbapi}
```

This command is used to set values for one or more configuration attributes. Attributes can be set on either the process level or instance level.

`set` commands are executed whether or not the cluster has been started. In a cluster that is not running, the MySQL Cluster Manager merely updates the configuration files. However, in a running cluster, the MySQL Cluster Manager in addition automatically performs any node restarts or rolling restarts (see [Performing a Rolling Restart of an NDB Cluster](#)) that are required to cause the attribute changes to take effect. However, since restart operations—particularly rolling restarts, where each process in the cluster must be stopped and restarted in turn—can take a great deal of time, it is preferable to make configuration changes before starting the cluster and putting it into use.

To set an attribute on the process level, use a `set` statement that contains an attribute assignment having the form `attribute_name:process_name=value`.

For example, to set `DataMemory` to 500 MB on the `ndbd` process level, so that the new value applies to all `ndbd` processes in the cluster, you can issue a `set` command containing the attribute assignment `DataMemory:ndbd=500M`, as shown here:

```
mcm> set DataMemory:ndbd=500M mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (5.68 sec)
```

To verify that the new setting is being used, you can issue the following `get` command:

```
mcm> get DataMemory mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value | Process1 | Id1  | Process2 | Id2  | Level  | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| DataMemory | 500M  | ndbd     | 1    | ndbd     | 2    | Instance |         |
+-----+-----+-----+-----+-----+-----+-----+
```


The `set` Command

```
| DataMemory | 500M | ndbd | 1 | | | Process | | |
| DataMemory | 500M | ndbd | 2 | | | Process | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.79 sec)
```



Note

For more information about this command, see [Section 4.5.1, “The `get` Command”](#).

To set an attribute for a specific process instance, include the process ID in the attribute assignment; the form of such an attribute assignment is `attribute_name:process_name:process_id=value`. For example, to set the `wait_timeout` attribute for the `mysqld` process that has process ID 50 to 200, you would issue a `set` command that contains the attribute assignment `wait_timeout:mysqld:50=200`, like this:

```
mcm> set wait_timeout:mysqld:50=200 mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Command result |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Cluster reconfigured successfully |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (6.18 sec)
```

You can verify that the setting has taken effect using an applicable `get` command:

```
mcm> get wait_timeout mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name | Value | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| wait_timeout | 200 | mysqld | 50 | | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.50 sec)
```

Attributes which are marked `Read only` cannot be set. Attempting to do so fails with an error, as shown here:

```
mcm> get :ndbd mycluster;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name | Value | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+-----+
| DataDir | /opt/c2data | ndbd | 1 | | | | |
| HostName | tonfisk | ndbd | 1 | | | | Read only |
| NodeId | 2 | ndbd | 1 | | | | Read only |
| DataDir | /opt/c3data | ndbd | 2 | | | | |
| HostName | grindval | ndbd | 2 | | | | Read only |
| NodeId | 3 | ndbd | 2 | | | | Read only |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (1.42 sec)
mcm> set HostName:ndbd:1=lax mycluster;
ERROR 6008 (00MGR): Config attribute HostName is read only and cannot be changed
```

However, you can set mandatory attributes, such as in the example shown previously in this section where the `DataDir` configuration attribute was set to a user-defined value.



Warning

The mandatory `NoOfReplicas` attribute must be set on the process level only. Attempting to set it on the instance level may leave the cluster, the MySQL Cluster Manager, or both in an unusable configuration.

Unlike the case with the `get` command, you cannot issue a `set` acting on a “global” scope—that is, you cannot, in a single attribute assignment, set a single value for an attribute such that the new attribute value applies to all processes regardless of process type, even if the attribute having that name can be applied to all process types. Nor can you specify multiple process types in a single attribute assignment. Attempting to do either of these things causes an error, as shown here:

```
mcm> set DataDir=/var/cluster-data mycluster;
ERROR 3 (00MGR): Illegal syntax

mcm> set DataDir:ndb_mgmd,ndbd,mysqld=/var/cluster-data mycluster;
ERROR 3 (00MGR): Illegal syntax
```

Instead, you must use a process-level attribute assignment for each process type. However, you are not necessarily required to issue a separate `set` command for each process type. Instead, you can also make multiple attribute assignments in a single `set` command, supplying the assignments as a comma-separated list. This `set` command assigns `/var/cdata` as the data directory (`DataDir`) for all MySQL Cluster processes in the cluster named `mycluster`:

```
mcm> set DataDir:ndb_mgmd=/var/cdata, \
        DataDir:ndbd=/var/cdata, \
        DataDir:mysqld=/var/cdata mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.66 sec)

mcm> get DataDir mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name      | Value      | Process1 | NodeId1 | Process2 | NodeId2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| DataDir   | /var/cdata | ndbmttd  | 1        |           |           |       |          |
| DataDir   | /var/cdata | ndbmttd  | 2        |           |           |       |          |
| DataDir   | /var/cdata | ndb_mgmd | 49       |           |           |       |          |
| datadir   | /var/cdata | mysqld   | 50       |           |           |       |          |
| datadir   | /var/cdata | mysqld   | 51       |           |           |       |          |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.08 sec)
```

As you can see from the `get` command just shown, the attribute assignments were successful, and took effect on the process level.



Note

In MySQL Cluster Manager, configuration attribute names are not case-sensitive. See [Case Sensitivity in String Searches](#) for more information about case-sensitivity issues in MySQL Cluster Manager.

Similarly, you cannot reference multiple process IDs in a single attribute assignment, even if they are processes of the same type; the following command does *not* work:

```
mcm> set DataMemory:ndbd:1,2=750M mycluster;
ERROR 3 (00MGR): Illegal syntax
```

Instead, you would need to use the following command:

```
mcm> set DataMemory:ndbd:1=750M,DataMemory:ndbd:2=750M mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
```

```
+-----+
1 row in set (7.70 sec)
```

(Of course, if these are the only two data nodes in `mycluster`, then the command `set DataMemory:ndbd=750M mycluster` also accomplishes the same task.)



Note

A few configuration attributes apply to connections between processes and so require you to refer to both processes in the course of setting them. In such cases, you must use a special process specification syntax; see [Setting TCP Connection Attributes](#), for information about how this is done.

You also cannot set values for multiple attributes in a single attribute assignment; this means that the following commands do *not* work:

```
mcm> set UndoDataBuffer=32M,UndoIndexBuffer=8M:ndbd mycluster;
ERROR 3 (00MGR): Illegal syntax

mcm> set DataMemory,IndexMemory:ndbd=1G mycluster;
ERROR 3 (00MGR): Illegal syntax
```

However, if you write a complete and valid attribute assignment for each attribute whose value you wish to update, you can rewrite these two commands so that they execute successfully, as shown here:

```
mcm> set UndoDataBuffer:ndbd=32M,UndoIndexBuffer:ndbd=8M mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (6.62 sec)

mcm> set DataMemory:ndbd=1G,IndexMemory:ndbd=1G mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.04 sec)
```

In fact, there is no reason that you cannot perform all four assignments in a single `set` command, using a list of four attribute assignments, like this:

```
mcm> set UndoDataBuffer:ndbd=32M,UndoIndexBuffer:ndbd=8M, \
      DataMemory:ndbd=1G, IndexMemory:ndbd=1G mycluster;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (6.24 sec)
```

However, it is a good idea not to perform too many attribute assignments in any single `set` command, since this makes it more difficult to spot errors.

On Windows, when setting attributes whose values contain paths (such as `DataDir`), you must replace any backslash characters in the path with forward slashes. Suppose that you want to use `C:\temp\node50` for the `tmpdir` attribute of the `mysqld` process having node ID 50 in a MySQL Cluster named `mycluster` which is running on Windows. The original value for this attribute can be seen using the appropriate `get` command:

```
mcm> get tmpdir mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name   | Value           | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| tmpdir | c:\c50data\tmp | mysql   | 50 |           |     |       |         |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.22 sec)
```

The correct `set` command to make the desired configuration change is shown here:

```
mcm> set tmpdir:mysql:50=c:/temp/node50 mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Cluster reconfigured successfully |
+-----+-----+
1 row in set (2.62 sec)
```

When you check the value using `get`—even though it was originally shown using backslashes—the forward slashes are used when displaying the new value:

```
mcm> get tmpdir mycluster;
+-----+-----+-----+-----+-----+-----+-----+
| Name   | Value           | Process1 | Id1 | Process2 | Id2 | Level | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| tmpdir | c:/temp/node50 | mysql   | 50 |           |     |       |         |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.22 sec)
```

However, if you try to use backslashes in the path when issuing the `set` command, the command fails:

```
mcm> set tmpdir:mysql:4=c:\temp\4 mycluster;
Outfile disabled.
ERROR:
Unknown command '\4'.
ERROR 6014 (00MGR): Path name for parameter tmpdir must be absolute.
The value 'c:mp4' is illegal.
```

Setting TCP Connection Attributes. For a few attributes that apply only when using TCP connections (such as the `SendBufferMemory` and `ReceiveBufferMemory` attributes), it is necessary to use a modified syntax for attribute value assignments. In this case, the attribute assignment contains two process specifications, one for each process type or instance to which the setting applies, joined with a plus sign (+). For the following example, consider the cluster named `mycluster2`, consisting of the processes shown here:

```
mcm> list processes mycluster2;
+-----+-----+-----+
| Id   | Name      | Host      |
+-----+-----+-----+
| 49   | ndb_mgmd | grindval  |
| 1    | ndbd     | tonfisk   |
| 2    | ndbd     | flundra   |
| 50   | mysql    | haj       |
| 51   | mysql    | torsk     |
+-----+-----+-----+
5 rows in set (0.16 sec)
```

(See [Section 4.6.3, “The list processes Command”](#), for more information about this command.)

TCP connection attributes are not shown in the output from the `get` command unless they have been set. This means that, prior to setting `SendBufferMemory` for the first time, you obtain an empty result if you try to retrieve its value, as shown here:

```
mcm> get SendBufferMemory mycluster2;
```

The `set` Command

```
Empty set (0.18 sec)

mcm> get --include-defaults SendBufferMemory mycluster2;
Empty set (0.93 sec)
```

To set the `SendBufferMemory` to 4 MB for all TCP connections between data nodes and SQL nodes, you can use the command shown here:

```
mcm> set SendBufferMemory:ndbd+mysqld=4M mycluster2;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (6.44 sec)
```

If you check the attribute's value afterwards using `get`, you can see that the value is applied to all possible connections between each of the two `ndbd` processes and each of the two `mysqld` processes in `mycluster2`, thus there are four rows in the output:

```
mcm> get SendBufferMemory mycluster2;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level  | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| SendBufferMemory | 4M    | ndbd     | 2   | mysqld   | 4   | Process |         |
| SendBufferMemory | 4M    | ndbd     | 2   | mysqld   | 5   | Process |         |
| SendBufferMemory | 4M    | ndbd     | 3   | mysqld   | 4   | Process |         |
| SendBufferMemory | 4M    | ndbd     | 3   | mysqld   | 5   | Process |         |
+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (1.63 sec)
```

To override this setting for only the connection between the data node with process ID 2 and the `mysqld` process (process ID 4), you can include the process ID in each of the two parts of the process specification, as shown here:

```
mcm> set SendBufferMemory:ndbd:2+mysqld:4=8M mycluster2;
+-----+
| Command result |
+-----+
| Cluster reconfigured successfully |
+-----+
1 row in set (7.95 sec)
```

When you check the result using a `get` command, you can see that the new setting applies on the instance level, and only to the connection between processes having IDs 2 and 4; the process-level setting made previously still applies to the remaining 3 connections:

```
mcm> get SendBufferMemory mycluster2;
+-----+-----+-----+-----+-----+-----+-----+
| Name          | Value | Process1 | Id1 | Process2 | Id2 | Level  | Comment |
+-----+-----+-----+-----+-----+-----+-----+
| SendBufferMemory | 8M    | ndbd     | 2   | mysqld   | 50  |        |         |
| SendBufferMemory | 4M    | ndbd     | 2   | mysqld   | 51  | Process |         |
| SendBufferMemory | 4M    | ndbd     | 3   | mysqld   | 50  | Process |         |
| SendBufferMemory | 4M    | ndbd     | 3   | mysqld   | 51  | Process |         |
+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.24 sec)
```

You cannot set a connection attribute on the process level in one part of the process specification (that is, for one end of the connection) and on the instance level in the other. Attempting to do so fails with an error, as shown here:

```
mcm> set SendBufferMemory:ndbd+mysqld:4=2M mycluster2;
ERROR 3 (00MGR): Illegal syntax
```

```
mcm> set SendBufferMemory:ndbd:2+mysqld=2M mycluster2;
ERROR 3 (00MGR): Illegal syntax
```

Setting Attributes for `mysqld` nodes. Before MySQL Cluster Manager 1.3.4, using the `set` command to set any attribute of a `mysqld` process always causes the process to restart. Since MySQL Cluster Manager 1.3.4, when a dynamic variable is set, `mcmd` sends out a SET GLOBAL statement instead to the `mysqld` to apply the value and also saves the value to the `mysqld` configuration file, so the value can be applied again at the next `mysqld` restart; however, an immediate restart is still triggered when a non-dynamic variable is set.

Setup for `mysqld` connection pooling. Enabling connection pooling for `mysqld` can be done by setting the `ndb-cluster-connection-pool` attribute to the desired number of connections, but also requires an extra step in creating the cluster.

Because the `mysqld` process attempts to make multiple connections to the cluster when connection pooling is enabled, the cluster must be configured with “spare” or “empty” connections. You can do this by adding (otherwise) unused `ndbapi` entries in the process_host list used in the `create cluster` command, as shown here:

```
mcm> create cluster -P mypackage
> -R ndb_mgmd@10.100.10.97,ndbd@10.100.10.98,ndbd@10.100.10.99, \
    mysqld@10.100.10.100,ndbapi@10.100.10.100, \
    ndbapi@10.100.10.100,ndbapi@10.100.10.100
> mycluster;
+-----+
| Command result |
+-----+
| Cluster created successfully |
+-----+
1 row in set (6.58 sec)
```

After this, you can use a `set` command like this one to set the size of the connection pool according to the number of excess connections available in the `config.ini` file:

```
mcm> set ndb_cluster_connection_pool:mysqld=4;
```

user attribute not supported for `mysqld`. Trying to set the `user` attribute for a `mysqld` process is not currently supported, and results in a warning being written to the MySQL Cluster Manager log.

4.6 MySQL Cluster Manager Process Commands

This section contains information about MySQL Cluster Manager client commands used to start and stop MySQL Cluster processes, and to determine which processes are currently running.

MySQL Cluster Manager, `ndb_mgm`, and starting or stopping processes. For a MySQL Cluster under MySQL Cluster Manager control, it is recommended *not* to use the `ndb_mgm` command-line client that comes with the MySQL Cluster distribution to perform operations that involve starting or stopping nodes. These include but are not limited to the following `ndb_mgm` client commands ([Commands in the NDB Cluster Management Client](#)):

- `START`
- `STOP`
- `RESTART`
- `SHUTDOWN`

4.6.1 The `add process` Command

```
add process {--processhosts=|-R }process_host_list
  [--set=attribute_assignment_list] cluster_name

process_host_list:
  process_name[:node_id]@host[,process_name@host[,...]]

process_name:
  {ndb_mgmd|ndbd|ndbmtid|mysqld|ndbapi}

attribute_assignment_list:
  attribute_assignment[,attribute_assignment][,...]

attribute_assignment:
  attribute_name:process_name[=value]
```

This command adds to an existing cluster one or more processes, which are specified using a `process_host_list` with the `--processhosts` option, the format of which is the same as that used with the `create cluster` command. Any hosts referenced in the list must be members of the site to which the cluster belongs. In addition, all hosts must be resolvable.

Any new processes added by this command must be started explicitly using the command `start process --added`.

For example, the following `add process` command adds two `mysqld` processes on hosts `tonfisk` and `flundra` to the cluster named `mycluster`:

```
mcm> add process --processhosts=mysqld@tonfisk,mysqld@flundra mycluster;
+-----+
| Command result |
+-----+
| Processes added successfully |
+-----+
1 row in set (2 min 10.39 sec)
```

You can also manually assign a node ID to the new process you are adding to the cluster by adding “:*node_ID*.” after the `process_name`. For MySQL Cluster Manager 1.3.3 and earlier, trying to manually assign node IDs less than 49 for `ndb_mgmd`, `mysqld`, or `ndbapi` fails with an error; the restriction, however, has been lifted since MySQL Cluster Manager 1.3.4. Nevertheless, you are still recommended to follow the best practice of reserving node ID 1 to 48 for data nodes. The following command adds two `ndbd` processes with node IDs 10 and 11 on hosts `tonfisk` and `flundra`, respectively, to `mycluster`:

```
mcm> add process --processhosts=ndbd:10@tonfisk,ndbd:11@flundra mycluster;
+-----+
| Command result |
+-----+
| Processes added successfully |
+-----+
1 row in set (2 min 13.40 sec)
```

Adding Free Processes

Using the `add process` command, you can add unmanaged `mysqld` processes, or `ndbapi` slots for `ndbapi` applications such as `ndb_restore`. To add an unmanaged `mysqld` process, prefix the hostname with the wildcard `*` (asterisk character):

```
mcm> add process --processhosts=mysqld@*tonfisk,mysqld@*flundra mycluster;
+-----+
| Command result |
+-----+
| Processes added successfully |
+-----+
1 row in set (2 min 3.14 sec)
```

To allow the unmanaged `mysqld` nodes to connect from any host, use the wildcard `*` (asterisk character) in place of the hostname or IP address:

```
mcm> add process --processhosts=mysqld@*,mysqld@* mycluster;
+-----+
| Command result |
+-----+
| Processes added successfully |
+-----+
1 row in set (2 min 3.14 sec)
```

The same applies to `ndbapi` slots for `ndbapi` applications such as `ndb_restore`: prefix the hostname with the wildcard character to limit connectivity to a specific host, or use only a wildcard, without hostname, to allow `ndbapi` applications from any host:

```
mcm> add process --processhosts=ndbapi@*tonfisk,ndbapi@* mycluster;
+-----+
| Command result |
+-----+
| Processes added successfully |
+-----+
1 row in set (2 min 8.13 sec)
```

Because “free” processes are not managed by MySQL Cluster Manager, there is no need to run the `start process --added` command after they have been successfully added to the cluster.

Using `add process` to Simplify `create cluster` Commands

Processes added before the cluster is started for the first time are started with the cluster. This makes it possible to use this command to break down what would otherwise be very long `create cluster` commands. Consider the following set of commands which creates and then starts a cluster named `mycluster`:

```
create cluster --processhosts=ndb_mgmd@host1,ndbd@host1,ndbd@host2, \
  mysqld@host3,mysqld@host4 mycluster;
start cluster mycluster;
```

The long `create cluster` command can be divided into a shorter (and more manageable) version of itself, plus several `add process` commands. This set of commands performs the same task as the previous set, creating `mycluster` with exactly the same processes and hosts as before, and then starting it:

```
create cluster --processhosts=ndb_mgmd@host1 mycluster;
add process --processhosts=ndbd@host1,ndbd@host2 mycluster;
add process --processhosts=mysqld@host3,mysqld@host4 mycluster;
start cluster mycluster;
```

Because `mycluster` was never started before the `ndbd` and `mysqld` processes were added to it using the `add process` commands shown, it is not necessary to use the `--added` option with the `start cluster` command in this case; all 5 processes are started.

A process that is added to a cluster that was created using `create cluster --import` and before the import takes place is added with status `import`, which means it cannot be started or stopped using `start process` or `stop process` before an import has taken place.

Configuring a New Process when Adding it

A newly added process inherits its configuration attribute settings from those in effect for its process type on the parent cluster, or assume the default settings for that process type if none apply. Existing attribute settings in the cluster must have process-level scope to be inherited by new processes added later;

instance-level settings set for existing process instances prior to adding any new ones do not apply to any of the added processes. (See [Configuration attributes](#), for more information about the scope of attribute settings.)

Inherited attribute settings can be overridden when adding processes; to do this, use the `add process` command's `--set` option. This option takes as its argument an attribute assignment list similar in format to that used with the `get` and `set` commands. Suppose that the current `ndbd` process-level setting in the cluster named `mycluster` for the `DataDir` attribute is `/home/users/ndb/cluster-data`, but you wish to add two new `ndbd` processes that use `/tmp/cluster/data` instead. You can do this using the following command:

```
mcm> add process --set=ndbd:DataDir=/tmp/cluster/data
> --processhosts=mysql@tonfisk,mysql@flundra
> mycluster;
```



Note

Unlike the way you use the `set` command, an equal sign (=) immediately following the `--set` option is required.

When setting attributes in this way which contain paths for processes running on Windows, you must replace any backslashes (\) used with forward slashes (/), just as with the `set` command. See [Setting Attributes Containing Paths on Windows \[125\]](#), for more information.

After a process has been added using `add process`, you can also use the `set` command to modify its configuration attribute settings (or specify additional ones) as you would with any other cluster process being managed with MySQL Cluster Manager.



Note

When IPv6-enabled Windows systems are used as MySQL Cluster hosts under MySQL Cluster Manager, you must reference these hosts using IPv4 addresses. Otherwise, MySQL Cluster Manager is unable to connect to the agent processes on those hosts. See [Section 5.1, “MySQL Cluster Manager Usage and Design Limitations”](#).

4.6.2 The `change process` Command

```
change process old_proc_type[:proc-id]=new_proc_type cluster_name

old_proc_type | new_proc_type:
{ndbd|ndbmt}
```

This command is used (MySQL Cluster NDB 7.0 and later) to change the process type for a given MySQL Cluster process or group of MySQL Cluster processes from one process type (*old-process-type*) to another process type (*new-process-type*).

Currently, the only two process types available for use with this command are `ndbd` and `ndbmt`. This means that `change process` can be used to change the data node process running on one or more data nodes from the single-threaded data node daemon (`ndbd`) to the multithreaded data node daemon (`ndbmt`) or vice versa.

By default, `change process` affects all data nodes running the *old-process-type*. By specifying an optional *process_id*, its action can be restricted to the data node having that process ID.

Suppose you have a cluster which is named `mycluster` and which has two data nodes using `ndbd` processes, as reflected in the output of the following `show status` command:

```
mcm> show status --process mycluster;
```

NodeId	Process	Host	Status	Nodegroup
49	ndb_mgmd	flundra	running	
1	ndbd	tonfisk	running	n/a
2	ndbd	grindval	running	n/a
50	mysqld	haj	running	
51	mysqld	torsk	running	
52	ndbapi	*	running	

```
6 rows in set (0.06 sec)
```

To change both data nodes to so that they use multithreaded (`ndbmttd`) processes, issue the command shown here, without any `process_id` specifier:

```
mcm> change process ndbd=ndbmttd mycluster;
```

```
Command result
```

```
Process changed successfully
```

```
1 row in set (2 min 17.51 sec)
```

After the command has executed, you can verify that both data nodes are now using `ndbmttd` by checking the output of the appropriate `show status` command, as shown here:

```
mcm> show status --process mycluster;
```

NodeId	Process	Host	Status	Nodegroup
49	ndb_mgmd	flundra	running	
1	ndbmttd	tonfisk	running	n/a
2	ndbmttd	grindval	running	n/a
50	mysqld	haj	running	
51	mysqld	torsk	running	
52	ndbapi	*	running	

```
6 rows in set (0.09 sec)
```



Note

The `change process` command can be used whether or not the cluster or the data node or data nodes to be changed are running. However, the command executes much more quickly if the data node or data nodes to be changed are not running. The next set of examples illustrates this.

It is possible (and sometimes desirable) to use `ndbd` and `ndbmttd` data node processes concurrently; thus, it is also possible using the `change process` command to change a single data node process from single-threaded to multithreaded, or from multithreaded to single-threaded. To do this, you must specify the data node process using its process ID.

First, we stop the cluster and verify that all processes are no longer running, as shown here:

```
mcm> stop cluster mycluster;
```

```
Command result
```

```
Cluster stopped successfully
```

```
1 row in set (22.93 sec)
```

```
mcm> show status --process mycluster;
```

The `change process` Command

```
+-----+-----+-----+-----+-----+
| NodeId | Process | Host   | Status | Nodegroup |
+-----+-----+-----+-----+-----+
| 49     | ndb_mgmd | flundra | stopped |           |
| 1      | ndbmttd  | tonfisk | stopped | n/a       |
| 2      | ndbmttd  | grindval | stopped | n/a       |
| 50     | mysqld   | haj     | stopped |           |
| 51     | mysqld   | torsk   | stopped |           |
| 52     | ndbapi   | *       | stopped |           |
+-----+-----+-----+-----+-----+
6 rows in set (0.05 sec)
```

The following command changes only the node having the process ID `2` from using the multithreaded data node daemon to the single-threaded version:

```
mcm> change process ndbmttd:2=ndbd mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Process changed successfully |
+-----+-----+
1 row in set (6.52 sec)
```

As you can see, `change process` operates much more quickly when the process to be changed is not running. As before, you can verify that the command succeeded using `show status`:

```
mcm> show status --process mycluster;
+-----+-----+-----+-----+-----+
| NodeId | Process | Host   | Status | Nodegroup |
+-----+-----+-----+-----+-----+
| 49     | ndb_mgmd | flundra | stopped |           |
| 1      | ndbmttd  | tonfisk | stopped | n/a       |
| 2      | ndbd     | grindval | stopped | n/a       |
| 50     | mysqld   | haj     | stopped |           |
| 51     | mysqld   | torsk   | stopped |           |
| 52     | ndbapi   | *       | stopped |           |
+-----+-----+-----+-----+-----+
6 rows in set (0.07 sec)
```

To complete the example, we start the cluster again, using `start cluster`, then change node number `2` back from `ndbd` (single-threaded) to `ndbmttd` (multithreaded) using `change process`, then verify the change using `show status`:

```
mcm> start cluster mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Cluster started successfully |
+-----+-----+
1 row in set (36.43 sec)

mcm> change process ndbd:2=ndbmttd mycluster;
+-----+-----+
| Command result |
+-----+-----+
| Process changed successfully |
+-----+-----+
1 row in set (2 min 10.41 sec)

mcm> show status --process mycluster;
+-----+-----+-----+-----+-----+
| NodeId | Process | Host   | Status | Nodegroup |
+-----+-----+-----+-----+-----+
| 49     | ndb_mgmd | flundra | running |           |
| 1      | ndbmttd  | tonfisk | running | n/a       |
```

The `list processes` Command

```
| 2 | ndbmtid | grindval | running | n/a |
| 50 | mysqld | haj | running | |
| 51 | mysqld | torsk | running | |
| 52 | ndbapi | * | running | |
+-----+-----+-----+-----+-----+
6 rows in set (0.11 sec)
```

You can see that it can require much less time to stop the cluster, change a data node process, and then start the cluster again than it is to change the process while the cluster is running. However, if you do this, the cluster is not available while it is stopped.

As noted previously, `change process` works only with `ndbd` and `ndbmtid` processes; attempting to use any other process type causes the command to fail with an error, as shown here:

```
mcm> change process ndb_mgmd=mysqld mycluster;
ERROR 7009 (OOMGR): Processes ndb_mgmd and mysqld are not interchangeable in this package
mcm> change process ndbd=mysqld mycluster;
ERROR 7009 (OOMGR): Processes ndbd and mysqld are not interchangeable in this package
```

4.6.3 The `list processes` Command

```
list processes cluster_name
```

This command displays all processes making up a given cluster. The following example demonstrates how to list all processes that are part of the cluster named `mycluster`:

```
mcm> list processes mycluster;
+-----+-----+-----+
| NodeId | Name   | Host   |
+-----+-----+-----+
| 49     | ndb_mgmd | flundra |
| 1      | ndbd    | tonfisk |
| 2      | ndbd    | grindval |
| 50     | mysqld  | haj     |
| 51     | mysqld  | torsk   |
| 52     | ndbapi  | *       |
+-----+-----+-----+
6 rows in set (0.03 sec)
```

The `cluster_name` argument is required. If this argument is omitted, the command fails with an error, as shown here:

```
mcm> list processes;
ERROR 6 (OOMGR): Illegal number of operands
```

4.6.4 The `start process` Command

```
start process [--initial|-i] process_id | --added} cluster_name
```

This command starts the MySQL Cluster process having the process ID `process_id` in the cluster named `cluster_name`. The status of the process to be started, as shown by `show status --process`, must be `added` or `stopped`.

This example demonstrates how to start the process having the process ID `1` belonging to the cluster `mycluster`:

```
mcm> start process 1 mycluster;
+-----+
| Command result |
+-----+
| Process started successfully |
+-----+
```

```
1 row in set (13.93 sec)
```

When the `--initial` option (short form: `-i`) is used, MySQL Cluster Manager starts a data node process with the `--initial` option, causing the data node to rebuild its file system.

Invoking this command with the `--added` option rather than with a process ID starts all nodes previously added to the cluster using `add process`, but not yet started.

You can obtain process IDs for all processes in a given cluster using `show status --process` or `list processes`. These are the same as the node IDs for these processes as shown in the output of other `mcm` client commands such as `get` or in the output of `ndb_mgm -e "show"` (see [ndb_mgm — The NDB Cluster Management Client](#)).

You cannot use this command to start a `mysqld` process in a cluster that is stopped or otherwise unavailable; in MySQL Cluster Manager 1.3.0 and later, trying to do so fails with an error. (Bug #17073352) This includes a cluster created for import where the import has not yet been completed (see [Section 4.4.1, “The create cluster Command”](#), and [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#)).

4.6.5 The `stop process` Command

```
stop process process_id cluster_name
```

This command stops the running MySQL Cluster process with the process ID `process_id` in the cluster named `cluster_name`. The status of the process as shown by `show status --process` must be `running`.

Suppose that the process ID of a data node in the cluster named `mycluster` is 3. Then this data node can be stopped as shown here:

```
mcm> stop process 3 mycluster;
+-----+
| Command result          |
+-----+
| Process stopped successfully |
+-----+
1 row in set (33.07 sec)
```

You can use `show status --process` or `list processes` to obtain process IDs for all processes in a given cluster.

In the event of a disk failure where MySQL Cluster Manager loses its manager directory (including its repository), the agent is able to recover information from other agents, but it does not actually control processes any longer, although it can detect them. This is due to the fact that the MySQL Cluster Manager agent cannot access the PID files. In this case, `stop process` no longer works, and you must kill such processes manually. Keep in mind that, if `StopOnError` is set to 0, the MySQL Cluster Manager agent restarts the data node process automatically; if `StopOnError` is 1 (the default), then you must execute the `start process` command manually.

This command does not work with processes in a cluster created for import where the import has not yet actually been completed. See [Section 4.4.1, “The create cluster Command”](#), and [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#), for more information.

4.6.6 The `remove process` Command

```
remove process [--removedirs] process_id_list cluster_name
process_id_list:
```

```
process_id[, process_id[, ...]]
```

This command removes permanently the processes in the `process_id_list` from the cluster named `cluster_name`. It provides a means to scale down a cluster offline.

If the `--removedirs` option is used, all data for the specified processes will be deleted.

The following restrictions apply when using this command:

1. The cluster must be in the status of `created` or `stopped`.
2. The processes to be removed must be in the status of `stopped`, `added`, or `import`.
3. The command cannot remove all processes from a cluster in the `created` status; at least one process must be left.
4. The command cannot remove all process of the same type from a cluster in the `stopped` status; at least one process must be left in the cluster for each type of nodes (management, data, and API).
5. The command cannot remove a data node that is in the `stopped` status if it is already a member of a node group (i.e., if it has ever been started and was fully functional).

You can use the `show status --process` or `list processes` command to obtain the process IDs for all the processes in a given cluster:

```
mcm> show status --process mycluster;
```

NodeId	Process	Host	Status	Nodegroup	Package
49	ndb_mgmd	flundra	added		mypackage
1	ndbmt	flundra	added	n/a	mypackage
2	ndbmt	flundra	added	n/a	mypackage
50	mysqld	flundra	added		mypackage
51	mysqld	flundra	added		mypackage
52	ndbapi	*	added		
53	ndbapi	*	added		

```
7 rows in set (0.03 sec)
```

The process IDs are the same as the node IDs for the processes shown in the output of the above or some other `mcm` client commands, or in the output of the `ndb_mgm -e "show"` command (see [ndb_mgm — The NDB Cluster Management Client](#)). In the above example, the SQL node with the process ID 50 in `mycluster` can be removed by the following command:

```
mcm> remove process 50 mycluster;
```

```
-----+
| Command result          |
+-----+
| Process removed successfully |
+-----+
1 row in set (0.48 sec)
```

And in this case, since the cluster was never started, we may also remove both data nodes:

```
mcm> remove process 1,2 mycluster;
```

```
-----+
| Command result          |
+-----+
| Process removed successfully |
+-----+
1 row in set (0.40 sec)
```

4.7 MySQL Cluster Manager Backup and Restore Commands

This section contains information about MySQL Cluster Manager client commands relating to backing up a MySQL Cluster and restoring it from backup.

4.7.1 The `abort backup` Command

```
abort backup --backupid=backup_id cluster_name
```

This command aborts a backup, having the specified *backup_id*, of cluster *cluster_name*. You can obtain a list of backups and their IDs known to this MySQL Cluster Manager instance using the `list backups`. If the backup is not actually in progress, the command has no effect.

4.7.2 The `backup cluster` Command

```
backup cluster
  [--backupid]
  [--snapshotstart | --snapshotend]
  [--waitstarted | --waitcompleted]
  cluster_name
```

This command creates a backup of the MySQL Cluster named *cluster_name*. `backup cluster` takes a backup of the cluster's NDB tables only; tables using other MySQL storage engines (such as InnoDB or MyISAM) are ignored.

By default, this command assigns a backup ID to each backup automatically; you can override this behavior by specifying a backup ID using the `--backupid` option.

The `--snapshotstart` option causes the backup to match the state of the cluster when the backup began.

The `--snapshotend` option causes the backup to reflect the state of the cluster when the backup was finished. If neither option is specified, the MySQL Cluster Manager client acts as though `--snapshotend` had been used.

When the `--waitstarted` option is used, the MySQL Cluster Manager client waits until the backup has started before returning control to the user, after which the user can check the backup process' status with the `show status` command and the `--backup` option.

Using `--waitcompleted` causes the MySQL Cluster Manager client to wait until the backup process is complete before returning control to the user. If neither of these options is specified, the client behaves as if `--waitcompleted` had been used.

```
mcm> backup cluster mycluster;
```

```
+-----+
| Command result |
+-----+
| Backup completed successfully |
+-----+
1 row in set (33.50 sec)
```

You can verify that the backup was performed by checking the output of `list backups`, as shown here:

```
mcm> list backups mycluster;
```

```
+-----+-----+-----+-----+-----+
| BackupId | NodeId | Host | Timestamp | Comment |
+-----+-----+-----+-----+-----+
| 1 | 1 | tonfisk | 2012-12-01 12:03:52 | |
+-----+-----+-----+-----+-----+
```

The `list backups` Command

```
| 1 | 2 | tonfisk | 2012-12-01 12:03:52 | | |
| 2 | 1 | tonfisk | 2012-12-04 12:05:15 | | |
| 2 | 2 | tonfisk | 2012-12-04 12:05:15 | | |
+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

Each row in the output represents a backup *image*—that is, a set of backup files specific to a given backup of a named cluster on a given data node. `Timestamp` values are UTC. By default, each image is stored as a directory named `/mcm_data/clusters/cluster_name/node_id/data/BACKUP/BACKUP-backup_id/` in the MySQL Cluster Manager installation directory.

It is possible to remove an unwanted backup from a given node by deleting this image directory and its contents. To remove a given backup completely, you must remove the corresponding image from each data node's `BACKUP` directory. You can do this as long as a backup or restore operation is not in progress; it is not necessary to stop the cluster or MySQL Cluster Manager agent prior to removing the images.

The `BackupId` is used with `abort backup` and `restore cluster`.

The `backup cluster` command was added in MySQL Cluster Manager 1.2.1.

4.7.3 The `list backups` Command

```
list backups cluster_name
```

This command lists all backups of the MySQL Cluster named `cluster_name` that are known to this instance of MySQL Cluster Manager. The output includes the backup and node ID as well as a UTC timestamp for each backup, as shown here:

```
mcm> list backups mycluster;
+-----+-----+-----+-----+-----+
| BackupId | NodeId | Host      | Timestamp           | Comment |
+-----+-----+-----+-----+-----+
| 1         | 1       | tonfisk   | 2012-12-01 12:03:52 |         |
| 1         | 2       | tonfisk   | 2012-12-01 12:03:52 |         |
| 2         | 1       | tonfisk   | 2012-12-04 12:05:15 |         |
| 2         | 2       | tonfisk   | 2012-12-04 12:05:15 |         |
+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

4.7.4 The `restore cluster` Command

```
restore cluster
  {--backupid=-I }backup_id
  [--disable-indexes|-x]
  [--disable-metadata|-M]
  [--epoch|-e]
  [--exclude-databases=db_name]
  [--exclude-intermediate-sql-tables]
  [--exclude-missing-columns]
  [--exclude-missing-tables]
  [--exclude-tables=db_name.tbl_name[,db_name.tbl_name][,...]]
  [--include-databases=db_name]
  [--include-tables=db_name.tbl_name[,db_name.tbl_name][,...]]
  [--lossy-conversions]
  [--no-binlog|-l]
  [--no-restore-disk-objects]
  [{--parallelism=-p }#]
  [--privilege-tables|-P]
  [--progress-frequency]
  [--rewrite-database]
  [--skip-broken-objects]
  [{--skip-nodeid=-s }id_list]
```



```
[--skip-table-check]
[--skip-unknown-objects]
cluster_name
```

This command restores a cluster from a backup having the specified backup ID (`--backupid` option; short form: `-I`) to the MySQL Cluster named `cluster_name`. In its simplest form, it can be used as shown here, to restore the cluster named `mycluster` to the state saved in the backup having backup ID 3:

```
mcm> restore cluster --backupid=3 mycluster;
+-----+
| Command result |
+-----+
| Restore completed successfully |
+-----+
1 row in set (18.60 sec)
```

If you are restoring an existing cluster to a known good state, you must wipe any existing data first. Stop the cluster using `stop cluster`, then restart it using `start cluster` with the `--initial` option, which causes the data node file systems to be cleared. (Note that Disk Data files must be removed manually.) Following this, you can restore the cluster from the desired backup using `restore cluster`.



Important

In order to restore a backup using `restore cluster`, the cluster must have an unused slot for an `ndbapi` process in its configuration. Otherwise, the command fails with the error `Unable to perform restore - no vacant ndbapi slots in config for cluster cluster_name`. See [Adding Free Processes](#), for information on how to add a free `ndbapi` slot to your cluster.

Additional options that can be employed with this command include:

`--disable-indexes` and `--disable-metadata`. To cause indexes to be ignored when restoring the table data, use the `--disable-indexes` option (short form: `-x`). Doing this can decrease the time required to restore a large data set, particularly where many indexes were in use. Similarly, you can cause metadata to be ignored during the restoration process by using the `--disable-metadata` option (short form: `-M`).

`--epoch`. When the `--epoch` option (short form: `-e`) is used, epoch information is restored to the cluster replication status table (`mysql.ndb_apply_status`), which can be useful for replicas in NDB Cluster replication.

`--exclude-databases` and `--exclude-tables`. Prevent one or more databases or tables from being restored using the options `--exclude-databases` and `--exclude-tables`. `--exclude-databases` takes a comma-delimited list of one or more databases which should not be restored. `--exclude-tables` takes a comma-delimited list of one or more tables (using the `database.table` format) which should not be restored. When `--exclude-databases` or `--exclude-tables` is used, only those databases or tables named by the option are excluded; all other databases and tables are restored.

`--exclude-missing-columns`. When this option is used, `restore cluster` ignores any columns missing from tables being restored as compared to the versions of those tables found in the backup.

`--exclude-missing-tables`. When this option is used, `restore cluster` ignores any tables from the backup that are not found in the target database.

`--exclude-intermediate-sql-tables[=TRUE|FALSE]`. When performing `ALTER TABLE` operations, `mysqld` creates intermediate tables (whose names are prefixed with `#sql-`). When `TRUE`, the

`--exclude-intermediate-sql-tables` option keeps `restore cluster` from restoring such tables that may have been left over from such operations. This option is `TRUE` by default.

`--include-databases` and `--include-tables`. Use the `--include-databases` option or the `--include-tables` option for restoring only specific databases or tables, respectively. `--include-databases` takes a comma-delimited list of databases to be restored. `--include-tables` takes a comma-delimited list of tables (in the `database.table` format) to be restored. When `--include-databases` or `--include-tables` is used, only those databases or tables named by the option are restored; all other databases and tables are excluded by `restore cluster`, and are not restored.

`--lossy-conversions`. Using `--lossy-conversions` allows lossy conversions of column values (type demotions or changes in sign) when restoring data from backup. With some exceptions, the rules governing demotion are the same as for MySQL replication; see [Replication of Columns Having Different Data Types](#), for information about specific type conversions currently supported by attribute demotion. `restore cluster` reports any truncation of data that it performs during lossy conversions once per attribute and column.

`--no-binlog`. The `--no-binlog` option (short form: `-l`) stops any SQL nodes (`mysqld` processes) in the cluster from writing data from the restore into their binary logs.

`--no-restore-disk-objects`. This option stops `restore cluster` from restoring any MySQL Cluster Disk Data objects, such as tablespaces and log file groups; see [NDB Cluster Disk Data Tables](#), for more information about these objects.

`--parallelism=#`. The `--parallelism` option (short form: `-p`) sets the maximum number of parallel transactions that the `restore cluster` command attempts to use. The default value is 128; the maximum is 1024, and the minimum is 1.

`--privilege-tables`. The `--privilege-tables` option (short form: `-P`) causes restoration of tables required for distributed grants (see [Distributed Privileges Using Shared Grant Tables](#)).

`--progress-frequency=N`. Print a status report each *N* seconds to a temporary stdout dump file `mcm` creates at `mcm_data/clusters/cluster_name/nodeid/tmp` while the backup is in progress. 0 (the default) causes no status reports to be printed. The maximum is 65535.

`--rewrite-database=old_dbname,new_dbname`. This option causes a database with the name `old_dbname` in the backup to be restored under the name `new_dbname`.

`--skip-nodeid`. The `--skip-nodeid` option (short form: `-s`) takes a comma-separated list of node IDs. The nodes whose IDs are listed may include of data nodes, SQL nodes, or both. Nodes having these IDs are skipped by the restoration process.

`--skip-broken-objects`. This option causes `restore cluster` to ignore corrupt tables while reading a backup, and to continue restoring any remaining tables (that are not also corrupted). Currently, the `--skip-broken-objects` option works only in the case of missing blob parts tables.

`--skip-table-check`. It is possible to restore data without restoring table metadata. The default behavior when doing this is for `restore cluster` to fail with an error if table data do not match the table schema; this can be overridden using the `--skip-table-check` option.

`--skip-unknown-objects`. This option causes `restore cluster` to ignore any schema objects it does not recognize while reading a backup. This can be used for restoring, for example, a backup made from a newer version of MySQL Cluster to an older version.

4.7.5 The `backup agents` Command

```
backup agents [--hosts=host_list] [site_name]

host_list:
  host[, host[, ...]]
```

This command backs up the configuration data for the `mcmd` agents on the hosts specified in `host_list` for the site named `site_name`. If no host names are specified, all agents of the site are backed up. If no `site_name` is given, only the agent that the `mcm` client is connected to is backed up.

The backup for each agent is created in a subfolder named `rep_backup/timestamp` under the agent repository (the `mcm_data` folder), with `timestamp` reflecting the time the backup began. If you want the backup to be at another place, create a soft link from `mcm_data/rep_backup` to your desired storage location.

Notice that the `backup agents` command works differently from the `backup cluster` command, which backs up cluster data; the `backup agents` command, on the other hand, backs up agent configuration data. Using together the backups created by both commands, you can restore not just the cluster, but the complete cluster-plus-manager setup. See [Section 3.7, “Backing Up and Restoring MySQL Cluster Manager Agents”](#) for more details about backing up and restoring `mcmd` agents.

4.8 MySQL Cluster Manager Cluster Importation Commands

This section contains descriptions of MySQL Cluster Manager commands used to perform operations connected with importing clusters into MySQL Cluster Manager. These operations include migration of cluster processes and (in MySQL Cluster Manager 1.3.1 and later) copying of configuration data.

4.8.1 The `import cluster` Command

```
import cluster [--dryrun|-y] cluster_name
```

This command imports a MySQL Cluster created independently of MySQL Cluster Manager into a cluster named `cluster_name` that has been created in MySQL Cluster Manager. You are strongly advised to create `cluster_name` using the `create cluster` command's `--import` option; see that command's description for more information about the `--import` option and its effects.

`import cluster` requires a single argument, the name of the cluster created using MySQL Cluster Manager (`cluster_name`) into which you wish to import a MySQL Cluster created externally into MySQL Cluster Manager and bring it under MySQL Cluster Manager control. The cluster named in the command must already exist in MySQL Cluster Manager.

`import cluster` also supports a `--dryrun` option. When this option is used, only the checks required for importation are performed against the existing cluster. This makes it possible to test a given configuration without actually placing any cluster processes under MCM control. `-y` is supported as a short form of this option beginning with MySQL Cluster Manager 1.3.1.

For more information about importing clusters into MySQL Cluster Manager, including examples, see [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#).

4.8.2 The `import config` Command

```
import config [--dryrun|-y] cluster_name
```

This command imports the configuration of an autonomous or “wild” cluster into the cluster named `cluster_name`.

`import config` requires a single argument, the name of the cluster created using MySQL Cluster Manager—that is, `cluster_name`—into which you wish to import the configuration of a MySQL Cluster

created externally into one created using MySQL Cluster Manager. The cluster named in the command must already exist in MySQL Cluster Manager; you are also strongly advised to use `create cluster --import` when creating `cluster_name`.

`import config` also supports a `--dryrun` option (short form: `-y`). When this option is used, only the checks required for importing the configuration data are performed. This makes it possible to perform tests without actually copying any of this data into a cluster under MCM control.

`import config` is available in MySQL Cluster Manager 1.3.1 and later.

For more information about importing clusters into MySQL Cluster Manager, including examples, see [Section 3.5, “Importing MySQL Clusters into MySQL Cluster Manager”](#).

Chapter 5 MySQL Cluster Manager Limitations and Known Issues

Table of Contents

5.1 MySQL Cluster Manager Usage and Design Limitations	143
5.2 MySQL Cluster Manager 1.3.6 Limitations Relating to the MySQL Server	144
5.3 MySQL Cluster Manager Limitations Relating to MySQL Cluster	144
5.4 Syntax and Related Issues in MySQL Cluster Manager	145

In this chapter we discuss limitations of and known issues in MySQL Cluster Manager version 1.3.6.

5.1 MySQL Cluster Manager Usage and Design Limitations

The limitations discussed in this section occur by intention or design in MySQL Cluster Manager 1.3.6. Some of these items may become obsolete in future versions; we will update this section accordingly if and as those changes come about.

change process command. Currently, the `change process` command can be used only to exchange an `ndbd` process for an `ndbmta` process, or the reverse. That is, in effect, it can be used only to switch a data node between a single-threaded process and a multithreaded process. It cannot be used for changing a cluster node's type (for example, you cannot change a data node to an SQL node, management node, or NDB API application node).

Because of this restriction, and due to the fact that multithreaded data nodes are not supported in MySQL Cluster NDB 6.3 and earlier, the `change process` command is useful only with clusters running MySQL Cluster NDB 7.0 or newer.

License keys and operating platforms. License keys are not required to use MySQL Cluster Manager 1.3.6.

Concurrent client sessions unsupported. Currently there is no negotiation or arbitration between multiple `mcm` clients. While it is possible to use the client from multiple locations, we do not support *concurrent* client sessions. You should be careful always to allow a command issued in one `mcm` client session to finish executing before issuing a new command in a different client session. This is true especially when using the `-B` or `--background` option with `mcm` client commands; see [Backgrounded commands and error handling](#).

IPv6 and host names (Windows). When IPv6 support is enabled on Windows systems, host names other than `localhost` are resolved using IPv6. When an IPv6-enabled Windows system is used as a MySQL Cluster host under MySQL Cluster Manager, you must reference it using its IPv4 address. Otherwise, `mcm` will be unable to connect to the agent process on that host.

This applies to host names used with the MySQL Cluster Manager client commands `create cluster`, `create site`, `add hosts`, `add package`, `delete package`, `stop agents`, and `add process`.

No removal of hosts (for MySQL Cluster Manager 1.3.4 and earlier). For MySQL Cluster Manager 1.3.4 and earlier, once a host is added to a management site using the `add hosts` command, it cannot be removed with MySQL Cluster Manager. For MySQL Cluster Manager 1.3.5 and later, a host can be removed from a site with the `remove hosts` command.

5.2 MySQL Cluster Manager 1.3.6 Limitations Relating to the MySQL Server

The limitations described in this section relate to functionality in the MySQL Server that is unsupported or reduced, or otherwise differs when using it with MySQL Cluster Manager.

Replication. Replication is currently not directly supported by MySQL Cluster Manager. See [Section 5.3, “MySQL Cluster Manager Limitations Relating to MySQL Cluster”](#), for more information.

Limited `mysqld` option modifier support. MySQL Cluster Manager does not recognize the `--loose`, `--maximum`, `--enable`, and `--disable` prefixes for `mysqld` options used as MySQL Cluster Manager configuration attributes (for a description of these modifiers, see [Program Option Modifiers](#)). For example, the command `set loose-skip-innodb:mysqld=true mycluster;` fails with the error `No such config variable loose-skip-innodb for process mysqld`.

The `--skip` option modifier is supported in some but not all cases, so that commands such as `set skip-innodb:mysqld=true mycluster;` and `set skip-grant-tables:mysqld=true mycluster;` can be used with MySQL Cluster Manager, while `set skip-column-names:mysqld=true mycluster;` cannot. (Bug #48559, Bug #47779)

Visibility of MySQL Cluster Manager `mysqld` attributes and MySQL server variables. Due in part to issues with mappings between `my.cnf` option names, `mysqld` command-line option names, and names of MySQL server variables, some MySQL server variables are not visible to MySQL Cluster Manager, or have different names. For example, the `storage_engine` server variable shown in the output of `SHOW VARIABLES` in the `mysql` client maps to the `default-storage-engine` configuration attribute used in the MySQL Cluster Manager `get` and `set` commands.

Dashes and underscores in MySQL option and variable names. When using the `mysql` client or other MySQL client applications, many MySQL system options and variables can be named using either dashes or underscores in their names. For example, you can use either `ndb_batch_size` or `ndb-batch-size` with the MySQL Server, and the variable is set correctly. This is not the case in MySQL Cluster Manager, where only the forms using underscores are accepted as attribute names. For example, assuming that `mycluster` is a viable cluster, the command `set ndb_batch_size:mysqld=65536 mycluster;` works to set the size of `ndb_batch_size` on all `mysqld` processes in the cluster, but `set ndb-batch-size:mysqld=65536 mycluster;` fails.

Dependencies between MySQL Cluster Manager `mysqld` attributes and MySQL server variables. MySQL Cluster Manager does not track dependencies between `mysqld` attributes (MySQL server options and system variables). That means MySQL Cluster Manager might have `mysqld` started successfully and report so, even though the server has ended up in a non-functional state because dependent attributes were set inconsistently. It is therefore a good idea for users to check the `mysqld` attributes before starting the node and the `mysql` log for status of the node after it has been started.

MySQL Cluster Manager `mysqld` attributes and MySQL user variables. MySQL user variables are not accessible as MySQL Cluster Manager configuration attributes.

Default MySQL data directory. Formerly, the default `tmpdir` for MySQL servers installed under MySQL Cluster Manager was `datadir/tmp`, which caused issues with the `mysql` client. This issue was resolved in MySQL Cluster Manager 1.2. MySQL Cluster Manager 1.3 (and later) uses `tmp` in the MySQL installation directory for this purpose.

5.3 MySQL Cluster Manager Limitations Relating to MySQL Cluster

This section describes limitations relating to MySQL Cluster functionality that is unsupported or curtailed by MySQL Cluster Manager 1.3.6.

MySQL Cluster Manager and replication. MySQL Cluster Manager currently does not provide any explicit support for MySQL Cluster Replication. However, you should still be able to perform manual setup of replication of a MySQL Cluster that is managed by MySQL Cluster Manager.

Backup and restore operations. MySQL Cluster Manager provides integrated backup and restore functionality. You can back up [NDB](#) databases and tables using the `mcm` client `backup cluster` command, and restore them using the `restore cluster` client command. MySQL Cluster Manager also supports restoration of distributed privileges.

You can also back up [NDB](#) databases and tables using the `ndb_mgm` client `START BACKUP` command, and restore them using the `ndb_restore` program; however MySQL Cluster Manager is not aware of backups that it was not employed to create. Both of the programs just mentioned are supplied with the MySQL NDB Cluster distribution.

**Note**

Backups of tables using storage engines other than [NDB](#), as well as of all other database objects which are not tables, cannot be made using MySQL Cluster Manager, and must be made using some other method, such as `mysqldump`.

Rolling restarts. Currently, all cluster nodes must be running in order to perform a rolling restart using MySQL Cluster Manager. However, MySQL Cluster itself requires only that at least one management server and all data nodes are running (in other words, any `mysqld` processes and any additional `ndb_mgmd` processes can be stopped). In such cases, you can perform the rolling restart manually, after stopping the MySQL Cluster Manager agent.

When making changes in configuration attributes only those nodes requiring a restart to make the change take effect are actually restarted. `ndbapi` nodes are never restarted by MySQL Cluster Manager.

Cluster Imports.

MySQL Cluster Manager will reject an import if it cannot access the process information of the cluster being imported. Therefore, the MySQL Cluster Manager agents must be run by a sufficiently privileged user—normally the same user that runs the cluster.

Cluster Reconfiguration. Cluster configuration updates (using the `set` or `reset` command) that would trigger a rolling restart of the nodes are not executed by MySQL Cluster Manager unless there are more than one data node defined for each node group; to perform such updates when the requirement is not met, a user should, using MySQL Cluster Manager, stop the cluster, use the `set` or `reset` command to change the cluster configurations, and then start the cluster again. If, however, your configuration changes require an [initial restart](#) of your cluster, you will need to backup the data, recreate your cluster from scratch with the new settings, and then restore your old data onto it.

5.4 Syntax and Related Issues in MySQL Cluster Manager

This section covers MySQL Cluster Manager issues relating to limitations in SQL and other syntax.

Backgrounded commands and error handling.

MySQL Cluster Manager client commands which are run in the background (that is, when they are invoked with the `--background` or `-B` option) do not issue any error messages in the client. If you run a command in the background, you must check the agent log or use the `show status` command to verify whether the backgrounded command was successful.

Appendix A Changes in MySQL Cluster Manager

MySQL Cluster Manager release notes are no longer published in the MySQL Cluster Manager User Manual.

Release notes for the changes in each release of MySQL Cluster Manager are located at [MySQL Cluster Manager 1.3 Release Notes](#).

Appendix B Attribute Summary Tables

Table of Contents

B.1 Management Node Configuration Parameters	149
B.2 Data Node Configuration Parameters	150
B.3 API Node Configuration Parameters	156
B.4 Other Node Configuration Parameters	156
B.5 MySQL Server Option and Variable Reference for MySQL Cluster	157

This appendix provides tables of configuration attributes, grouped according to their process type or by the section of the MySQL Cluster configuration file in which they appear. This information is current for MySQL Cluster NDB 7.3.

Each table provides the following information:

- *Name*: The name of the attribute. The name of the attribute is linked to the attribute's full description in the online MySQL Cluster documentation.
- *Type/Units*: The data type or unit by which the attribute is measured.
- *Range*: The default value of the attribute if not set by the user, and the minimum and maximum values which can be set for the attribute.
- *Restart Type*: The type of restart required for a change in value in this attribute to be applied in a running MySQL Cluster. The restart type is indicated in this column by an **N** for a node restart, or an **S** for a system restart. *Data node attributes*: The presence of an **I** in this column indicates that a data node must be restarted using the `--initial` option for a change to take effect.

Attributes having restart type **N** can be changed using a rolling restart of the cluster, and thus can be changed at any time, even if the cluster is running. Changing an attribute whose restart type is **S** requires a complete shutdown of all cluster nodes, followed by a restart of the nodes once all of them have been stopped. Currently, such attributes can be set only before starting a cluster for the first time.

B.1 Management Node Configuration Parameters

- `ArbitrationDelay`: When asked to arbitrate, arbitrator waits this long before voting (milliseconds).
- `ArbitrationRank`: If 0, then management node is not arbitrator. Kernel selects arbitrators in order 1, 2.
- `DataDir`: Data directory for this node.
- `ExecuteOnComputer`: String referencing earlier defined COMPUTER.
- `ExtraSendBufferMemory`: Memory to use for send buffers in addition to any allocated by `TotalSendBufferMemory` or `SendBufferMemory`. Default (0) allows up to 16MB.
- `HeartbeatIntervalMgmdMgmd`: Time between management-node-to-management-node heartbeats; connection between management nodes is considered lost after 3 missed heartbeats.
- `HeartbeatThreadPriority`: Set heartbeat thread policy and priority for management nodes; see manual for allowed values.

- **HostName**: Host name or IP address for this management node.
- **Id**: Number identifying management node. Now deprecated; use **NodeId** instead.
- **LogDestination**: Where to send log messages: console, system log, or specified log file.
- **MaxNoOfSavedEvents**: Not used.
- **NodeId**: Number uniquely identifying management node among all nodes in cluster.
- **PortNumber**: Port number to send commands to and fetch configuration from management server.
- **PortNumberStats**: Port number used to get statistical information from management server.
- **TotalSendBufferMemory**: Total memory to use for all transporter send buffers.
- **wan**: Use WAN TCP setting as default.

B.2 Data Node Configuration Parameters

- **Arbitration**: How arbitration should be performed to avoid split-brain issues in event of node failure.
- **ArbitrationTimeout**: Maximum time (milliseconds) database partition waits for arbitration signal.
- **BackupDataBufferSize**: Default size of databuffer for backup (in bytes).
- **BackupDataDir**: Path to where to store backups. Note that string '/BACKUP' is always appended to this setting, so that *effective* default is `FileSystemPath/BACKUP`.
- **BackupDiskWriteSpeedPct**: Sets percentage of data node's allocated maximum write speed (`MaxDiskWriteSpeed`) to reserve for LCPs when starting backup.
- **BackupLogBufferSize**: Default size of log buffer for backup (in bytes).
- **BackupMaxWriteSize**: Maximum size of file system writes made by backup (in bytes).
- **BackupMemory**: Total memory allocated for backups per node (in bytes).
- **BackupReportFrequency**: Frequency of backup status reports during backup in seconds.
- **BackupWriteSize**: Default size of file system writes made by backup (in bytes).
- **BatchSizePerLocalScan**: Used to calculate number of lock records for scan with hold lock.
- **BuildIndexThreads**: Number of threads to use for building ordered indexes during system or node restart. Also applies when running `ndb_restore --rebuild-indexes`. Setting this parameter to 0 disables multithreaded building of ordered indexes.
- **CompressedBackup**: Use `zlib` to compress backups as they are written.
- **CompressedLCP**: Write compressed LCPs using `zlib`.
- **ConnectCheckIntervalDelay**: Time between data node connectivity check stages. Data node is considered suspect after 1 interval and dead after 2 intervals with no response.
- **CrashOnCorruptedTuple**: When enabled, forces node to shut down whenever it detects corrupted tuple.
- **DataDir**: Data directory for this node.

- **DataMemory**: Number of bytes on each data node allocated for storing data; subject to available system RAM and size of IndexMemory.
- **DefaultHashMapSize**: Set size (in buckets) to use for table hash maps. Three values are supported: 0, 240, and 3840..
- **DictTrace**: Enable DBDICT debugging; for NDB development.
- **DiskCheckpointSpeed**: Bytes allowed to be written by checkpoint, per second.
- **DiskCheckpointSpeedInRestart**: Bytes allowed to be written by checkpoint during restart, per second.
- **DiskIOThreadPool**: Number of unbound threads for file access, applies to disk data only.
- **Diskless**: Run without using disk.
- **DiskPageBufferEntries**: Memory to allocate in DiskPageBufferMemory; very large disk transactions may require increasing this value.
- **DiskPageBufferMemory**: Number of bytes on each data node allocated for disk page buffer cache.
- **DiskSyncSize**: Amount of data written to file before synch is forced.
- **EventLogBufferSize**: Size of circular buffer for NDB log events within data nodes.
- **ExecuteOnComputer**: String referencing earlier defined COMPUTER.
- **ExtraSendBufferMemory**: Memory to use for send buffers in addition to any allocated by TotalSendBufferMemory or SendBufferMemory. Default (0) allows up to 16MB.
- **FileSystemPath**: Path to directory where data node stores its data (directory must exist).
- **FileSystemPathDataFiles**: Path to directory where data node stores its Disk Data files. Default value is FilesystemPathDD, if set; otherwise, FilesystemPath is used if it is set; otherwise, value of DataDir is used.
- **FileSystemPathDD**: Path to directory where data node stores its Disk Data and undo files. Default value is FilesystemPath, if set; otherwise, value of DataDir is used.
- **FileSystemPathUndoFiles**: Path to directory where data node stores its undo files for Disk Data. Default value is FilesystemPathDD, if set; otherwise, FilesystemPath is used if it is set; otherwise, value of DataDir is used.
- **FragmentLogFileSize**: Size of each redo log file.
- **HeartbeatIntervalDbApi**: Time between API node-data node heartbeats. (API connection closed after 3 missed heartbeats).
- **HeartbeatIntervalDbDb**: Time between data node-to-data node heartbeats; data node considered dead after 3 missed heartbeats.
- **HeartbeatOrder**: Sets order in which data nodes check each others' heartbeats for determining whether given node is still active and connected to cluster. Must be zero for all data nodes or distinct nonzero values for all data nodes; see documentation for further guidance.
- **HostName**: Host name or IP address for this data node.
- **Id**: Number identifying data node. Now deprecated; use NodeId instead.

- **IndexMemory**: Number of bytes on each data node allocated for storing indexes; subject to available system RAM and size of DataMemory.
- **IndexStatAutoCreate**: Enable/disable automatic statistics collection when indexes are created.
- **IndexStatAutoUpdate**: Monitor indexes for changes and trigger automatic statistics updates.
- **IndexStatSaveScale**: Scaling factor used in determining size of stored index statistics.
- **IndexStatSaveSize**: Maximum size in bytes for saved statistics per index.
- **IndexStatTriggerPct**: Threshold percent change in DML operations for index statistics updates. Value is scaled down by IndexStatTriggerScale.
- **IndexStatTriggerScale**: Scale down IndexStatTriggerPct by this amount, multiplied by base 2 logarithm of index size, for large index. Set to 0 to disable scaling.
- **IndexStatUpdateDelay**: Minimum delay between automatic index statistics updates for given index. 0 means no delay.
- **InitFragmentLogFiles**: Initialize fragment logfiles (sparse/full).
- **InitialLogFileGroup**: Describes log file group that is created during initial start. See documentation for format.
- **InitialNoOfOpenFiles**: Initial number of files open per data node. (One thread is created per file).
- **InitialTablespace**: Describes tablespace that is created during initial start. See documentation for format.
- **LateAlloc**: Allocate memory after connection to management server has been established.
- **LcpScanProgressTimeout**: Maximum time that local checkpoint fragment scan can be stalled before node is shut down to ensure systemwide LCP progress. Use 0 to disable.
- **LockExecuteThreadToCPU**: Comma-delimited list of CPU IDs.
- **LockMaintThreadsToCPU**: CPU ID indicating which CPU runs maintenance threads.
- **LockPagesInMainMemory**: 0=disable locking, 1=lock after memory allocation, 2=lock before memory allocation.
- **LogLevelCheckpoint**: Log level of local and global checkpoint information printed to stdout.
- **LogLevelCongestion**: Level of congestion information printed to stdout.
- **LogLevelConnection**: Level of node connect/disconnect information printed to stdout.
- **LogLevelError**: Transporter, heartbeat errors printed to stdout.
- **LogLevelInfo**: Heartbeat and log information printed to stdout.
- **LogLevelNodeRestart**: Level of node restart and node failure information printed to stdout.
- **LogLevelShutdown**: Level of node shutdown information printed to stdout.
- **LogLevelStartup**: Level of node startup information printed to stdout.
- **LogLevelStatistic**: Level of transaction, operation, and transporter information printed to stdout.

- [LongMessageBuffer](#): Number of bytes allocated on each data node for internal long messages.
- [MaxAllocate](#): Maximum size of allocation to use when allocating memory for tables.
- [MaxBufferedEpochs](#): Allowed number of epochs that subscribing node can lag behind (unprocessed epochs). Exceeding causes lagging subscribers to be disconnected.
- [MaxBufferedEpochBytes](#): Total number of bytes allocated for buffering epochs.
- [MaxDiskWriteSpeed](#): Maximum number of bytes per second that can be written by LCP and backup when no restarts are ongoing.
- [MaxDiskWriteSpeedOtherNodeRestart](#): Maximum number of bytes per second that can be written by LCP and backup when another node is restarting.
- [MaxDiskWriteSpeedOwnRestart](#): Maximum number of bytes per second that can be written by LCP and backup when this node is restarting.
- [MaxDMLOperationsPerTransaction](#): Limit size of transaction; aborts transaction if it requires more than this many DML operations. Set to 0 to disable.
- [MaxLCPStartDelay](#): Time in seconds that LCP polls for checkpoint mutex (to allow other data nodes to complete metadata synchronization), before putting itself in lock queue for parallel recovery of table data.
- [MaxNoOfAttributes](#): Suggests total number of attributes stored in database (sum over all tables).
- [MaxNoOfConcurrentIndexOperations](#): Total number of index operations that can execute simultaneously on one data node.
- [MaxNoOfConcurrentOperations](#): Maximum number of operation records in transaction coordinator.
- [MaxNoOfConcurrentScans](#): Maximum number of scans executing concurrently on data node.
- [MaxNoOfConcurrentSubOperations](#): Maximum number of concurrent subscriber operations.
- [MaxNoOfConcurrentTransactions](#): Maximum number of transactions executing concurrently on this data node, total number of transactions that can be executed concurrently is this value times number of data nodes in cluster.
- [MaxNoOfFiredTriggers](#): Total number of triggers that can fire simultaneously on one data node.
- [MaxNoOfLocalOperations](#): Maximum number of operation records defined on this data node.
- [MaxNoOfLocalScans](#): Maximum number of fragment scans in parallel on this data node.
- [MaxNoOfOpenFiles](#): Maximum number of files open per data node.(One thread is created per file).
- [MaxNoOfOrderedIndexes](#): Total number of ordered indexes that can be defined in system.
- [MaxNoOfSavedMessages](#): Maximum number of error messages to write in error log and maximum number of trace files to retain.
- [MaxNoOfSubscribers](#): Maximum number of subscribers.
- [MaxNoOfSubscriptions](#): Maximum number of subscriptions (default 0 = MaxNoOfTables).
- [MaxNoOfTables](#): Suggests total number of NDB tables stored in database.
- [MaxNoOfTriggers](#): Total number of triggers that can be defined in system.

- **MaxNoOfUniqueHashIndexes**: Total number of unique hash indexes that can be defined in system.
- **MaxParallelCopyInstances**: Number of parallel copies during node restarts. Default is 0, which uses number of LDMs on both nodes, to maximum of 16.
- **MaxParallelScansPerFragment**: Maximum number of parallel scans per fragment. Once this limit is reached, scans are serialized.
- **MaxStartFailRetries**: Maximum retries when data node fails on startup, requires StopOnError = 0. Setting to 0 causes start attempts to continue indefinitely.
- **MemReportFrequency**: Frequency of memory reports in seconds; 0 = report only when exceeding percentage limits.
- **MinDiskWriteSpeed**: Minimum number of bytes per second that can be written by LCP and backup.
- **MinFreePct**: Percentage of memory resources to keep in reserve for restarts.
- **NodeGroup**: Node group to which data node belongs; used only during initial start of cluster.
- **NodeId**: Number uniquely identifying data node among all nodes in cluster.
- **NoOfFragmentLogFiles**: Number of 16 MB redo log files in each of 4 file sets belonging to data node.
- **NoOfReplicas**: Number of copies of all data in database.
- **Numa**: (Linux only; requires libnuma) Controls NUMA support. Setting to 0 permits system to determine use of interleaving by data node process; 1 means that it is determined by data node.
- **ODirect**: Use O_DIRECT file reads and writes when possible.
- **RealtimeScheduler**: When true, data node threads are scheduled as real-time threads. Default is false.
- **RedoBuffer**: Number of bytes on each data node allocated for writing redo logs.
- **RedoOverCommitCounter**: When RedoOverCommitLimit has been exceeded this many times, transactions are aborted, and operations are handled as specified by DefaultOperationRedoProblemAction.
- **RedoOverCommitLimit**: Each time that flushing current redo buffer takes longer than this many seconds, number of times that this has happened is compared to RedoOverCommitCounter.
- **ReservedSendBufferMemory**: This parameter is present in NDB code but is not enabled.
- **RestartOnErrorInsert**: Control type of restart caused by inserting error (when StopOnError is enabled).
- **SchedulerExecutionTimer**: Number of microseconds to execute in scheduler before sending.
- **SchedulerResponsiveness**: Set NDB scheduler response optimization 0-10; higher values provide better response time but lower throughput.
- **SchedulerSpinTimer**: Number of microseconds to execute in scheduler before sleeping.
- **ServerPort**: Port used to set up transporter for incoming connections from API nodes.
- **SharedGlobalMemory**: Total number of bytes on each data node allocated for any use.
- **StartFailRetryDelay**: Delay in seconds after start failure prior to retry; requires StopOnError = 0.

- `StartFailureTimeout`: Milliseconds to wait before terminating. (0=Wait forever).
- `StartNoNodeGroupTimeout`: Time to wait for nodes without nodegroup before trying to start (0=forever).
- `StartPartialTimeout`: Milliseconds to wait before trying to start without all nodes. (0=Wait forever).
- `StartPartitionedTimeout`: Milliseconds to wait before trying to start partitioned. (0=Wait forever).
- `StartupStatusReportFrequency`: Frequency of status reports during startup.
- `StopOnError`: When set to 0, data node automatically restarts and recovers following node failures.
- `StringMemory`: Default size of string memory (0 to 100 = % of maximum, 101+ = actual bytes).
- `TcpBind_INADDR_ANY`: Bind IP_ADDR_ANY so that connections can be made from anywhere (for autogenerated connections).
- `TimeBetweenEpochs`: Time between epochs (synchronization used for replication).
- `TimeBetweenEpochsTimeout`: Timeout for time between epochs. Exceeding causes node shutdown.
- `TimeBetweenGlobalCheckpoints`: Time between group commits of transactions to disk.
- `TimeBetweenGlobalCheckpointsTimeout`: Minimum timeout for group commit of transactions to disk.
- `TimeBetweenInactiveTransactionAbortCheck`: Time between checks for inactive transactions.
- `TimeBetweenLocalCheckpoints`: Time between taking snapshots of database (expressed in base-2 logarithm of bytes).
- `TimeBetweenWatchDogCheck`: Time between execution checks inside data node.
- `TimeBetweenWatchDogCheckInitial`: Time between execution checks inside data node (early start phases when memory is allocated).
- `TotalSendBufferMemory`: Total memory to use for all transporter send buffers..
- `TransactionBufferMemory`: Dynamic buffer space (in bytes) for key and attribute data allocated for each data node.
- `TransactionDeadlockDetectionTimeout`: Time transaction can spend executing within data node. This is time that transaction coordinator waits for each data node participating in transaction to execute request. If data node takes more than this amount of time, transaction is aborted.
- `TransactionInactiveTimeout`: Milliseconds that application waits before executing another part of transaction. This is time transaction coordinator waits for application to execute or send another part (query, statement) of transaction. If application takes too much time, then transaction is aborted. Timeout = 0 means that application never times out.
- `TwoPassInitialNodeRestartCopy`: Copy data in 2 passes during initial node restart, which enables multithreaded building of ordered indexes for such restarts.
- `UndoDataBuffer`: Number of bytes on each data node allocated for writing data undo logs.
- `UndoIndexBuffer`: Number of bytes on each data node allocated for writing index undo logs.
- `MaxNoOfExecutionThreads`: For ndbmt only, specify maximum number of execution threads.

- `NoOfFragmentLogParts`: Number of redo log file groups belonging to this data node.
- `ThreadConfig`: Used for configuration of multithreaded data nodes (ndbmttd). Default is empty string; see documentation for syntax and other information.

B.3 API Node Configuration Parameters

- `ApiVerbose`: Enable NDB API debugging; for NDB development.
- `ArbitrationDelay`: When asked to arbitrate, arbitrator waits this many milliseconds before voting.
- `ArbitrationRank`: If 0, then API node is not arbitrator. Kernel selects arbitrators in order 1, 2.
- `AutoReconnect`: Specifies whether an API node should reconnect fully when disconnected from cluster.
- `BatchByteSize`: Default batch size in bytes.
- `BatchSize`: Default batch size in number of records.
- `ConnectBackoffMaxTime`: Specifies longest time in milliseconds (~100ms resolution) to allow between connection attempts to any given data node by this API node. Excludes time elapsed while connection attempts are ongoing, which in worst case can take several seconds. Disable by setting to 0. If no data nodes are currently connected to this API node, `StartConnectBackoffMaxTime` is used instead.
- `ConnectionMap`: Specifies which data nodes to connect.
- `DefaultHashMapSize`: Set size (in buckets) to use for table hash maps. Three values are supported: 0, 240, and 3840.
- `DefaultOperationRedoProblemAction`: How operations are handled in event that `RedoOverCommitCounter` is exceeded.
- `ExecuteOnComputer`: String referencing earlier defined COMPUTER.
- `ExtraSendBufferMemory`: Memory to use for send buffers in addition to any allocated by `TotalSendBufferMemory` or `SendBufferMemory`. Default (0) allows up to 16MB.
- `HeartbeatThreadPriority`: Set heartbeat thread policy and priority for API nodes; see manual for allowed values.
- `HostName`: Host name or IP address for this SQL or API node.
- `Id`: Number identifying MySQL server or API node (Id). Now deprecated; use `NodeId` instead.
- `MaxScanBatchSize`: Maximum collective batch size for one scan.
- `NodeId`: Number uniquely identifying SQL node or API node among all nodes in cluster.
- `StartConnectBackoffMaxTime`: Same as `ConnectBackoffMaxTime` except that this parameter is used in its place if no data nodes are connected to this API node.
- `TotalSendBufferMemory`: Total memory to use for all transporter send buffers.
- `wan`: Use WAN TCP setting as default.

B.4 Other Node Configuration Parameters

- `HostName`: Host name or IP address of this computer.

- [Id](#): Unique identifier for this computer.
- [Checksum](#): If checksum is enabled, all signals between nodes are checked for errors.
- [Group](#): Used for group proximity; smaller value is interpreted as being closer.
- [NodeId1](#): ID of node (data node, API node, or management node) on one side of connection.
- [NodeId2](#): ID of node (data node, API node, or management node) on one side of connection.
- [NodeIdServer](#): Set server side of TCP connection.
- [OverloadLimit](#): When more than this many unsent bytes are in send buffer, connection is considered overloaded.
- [PortNumber](#): Port used for TCP transporter.
- [Proxy](#): .
- [ReceiveBufferMemory](#): Bytes of buffer for signals received by this node.
- [SendBufferMemory](#): Bytes of TCP buffer for signals sent from this node.
- [SendSignalId](#): Sends ID in each signal. Used in trace files. Defaults to true in debug builds.
- [TCP_MAXSEG_SIZE](#): Value used for TCP_MAXSEG.
- [TCP_RCV_BUF_SIZE](#): Value used for SO_RCVBUF.
- [TCP_SND_BUF_SIZE](#): Value used for SO_SNDBUF.
- [TcpBind_INADDR_ANY](#): Bind InAddrAny instead of host name for server part of connection.
- [Checksum](#): If checksum is enabled, all signals between nodes are checked for errors.
- [Group](#): Used for group proximity; smaller value is interpreted as being closer.
- [NodeId1](#): ID of node (data node, API node, or management node) on one side of connection.
- [NodeId2](#): ID of node (data node, API node, or management node) on one side of connection.
- [NodeIdServer](#): Set server side of SHM connection.
- [OverloadLimit](#): When more than this many unsent bytes are in send buffer, connection is considered overloaded.
- [PortNumber](#): Port used for SHM transporter.
- [SendSignalId](#): Sends ID in each signal. Used in trace files.
- [ShmKey](#): Shared memory key; when set to 1, this is calculated by NDB.
- [ShmSize](#): Size of shared memory segment.
- [Signum](#): Signal number to be used for signalling.

B.5 MySQL Server Option and Variable Reference for MySQL Cluster

- [Com_show_ndb_status](#): Count of SHOW NDB STATUS statements.

- `create_old_temporals`: Use pre-5.6.4 storage format for temporal types when creating tables. Intended for use in replication and upgrades/downgrades between NDB 7.2 and NDB 7.3/7.4.
- `Handler_discover`: Number of times that tables have been discovered.
- `ndb-batch-size`: Size (in bytes) to use for NDB transaction batches.
- `ndb-blob-read-batch-bytes`: Specifies size in bytes that large BLOB reads should be batched into. 0 = no limit.
- `ndb-blob-write-batch-bytes`: Specifies size in bytes that large BLOB writes should be batched into. 0 = no limit.
- `ndb-cluster-connection-pool`: Number of connections to cluster used by MySQL.
- `ndb-connectstring`: Address of NDB management server distributing configuration information for this cluster.
- `ndb-deferred-constraints`: Specifies that constraint checks on unique indexes (where these are supported) should be deferred until commit time. Not normally needed or used; for testing purposes only.
- `ndb-distribution`: Default distribution for new tables in NDBCLUSTER (KEYHASH or LINHASH, default is KEYHASH).
- `ndb-log-apply-status`: Cause MySQL server acting as replica to log `mysql.ndb_apply_status` updates received from its immediate source in its own binary log, using its own server ID. Effective only if server is started with `--ndbcluster` option.
- `ndb-log-empty-epochs`: When enabled, causes epochs in which there were no changes to be written to `ndb_apply_status` and `ndb_binlog_index` tables, even when `--log-slave-updates` is enabled.
- `ndb-log-empty-update`: When enabled, causes updates that produced no changes to be written to `ndb_apply_status` and `ndb_binlog_index` tables, even when `--log-slave-updates` is enabled.
- `ndb-log-exclusive-reads`: Log primary key reads with exclusive locks; allow conflict resolution based on read conflicts.
- `ndb-log-fail-terminate`: Terminate `mysqld` process if complete logging of all found row events is not possible.
- `ndb-log-orig`: Log originating server id and epoch in `mysql.ndb_binlog_index` table.
- `ndb-log-transaction-id`: Write NDB transaction IDs in binary log. Requires `--log-bin-v1-events=OFF`.
- `ndb-log-update-as-write`: Toggles logging of updates on source between updates (OFF) and writes (ON).
- `ndb-mgmd-host`: Set host (and port, if desired) for connecting to management server.
- `ndb-nodeid`: NDB Cluster node ID for this MySQL server.
- `ndb-transid-mysql-connection-map`: Enable or disable `ndb_transid_mysql_connection_map` plugin; that is, enable or disable `INFORMATION_SCHEMA` table having that name.
- `ndb-wait-connected`: Time (in seconds) for MySQL server to wait for connection to cluster management and data nodes before accepting MySQL client connections.
- `ndb-wait-setup`: Time (in seconds) for MySQL server to wait for NDB engine setup to complete.

- [Ndb_api_adaptive_send_deferred_count](#): Number of adaptive send calls not actually sent by this MySQL Server (SQL node).
- [Ndb_api_adaptive_send_deferred_count_slave](#): Number of adaptive send calls not actually sent in this client session.
- [Ndb_api_adaptive_send_deferred_count_slave](#): Number of adaptive send calls not actually sent by this replica.
- [Ndb_api_adaptive_send_forced_count](#): Number of adaptive sends with forced-send set sent by this MySQL Server (SQL node).
- [Ndb_api_adaptive_send_forced_count_session](#): Number of adaptive sends with forced-send set in this client session.
- [Ndb_api_adaptive_send_forced_count_slave](#): Number of adaptive sends with forced-send set sent by this replica.
- [Ndb_api_adaptive_send_unforced_count](#): Number of adaptive sends without forced-send sent by this MySQL Server (SQL node).
- [Ndb_api_adaptive_send_unforced_count_session](#): Number of adaptive sends without forced-send in this client session.
- [Ndb_api_adaptive_send_unforced_count_slave](#): Number of adaptive sends without forced-send sent by this replica.
- [Ndb_api_bytes_received_count](#): Quantity of data (in bytes) received from data nodes by this MySQL Server (SQL node).
- [Ndb_api_bytes_received_count_session](#): Quantity of data (in bytes) received from data nodes in this client session.
- [Ndb_api_bytes_received_count_slave](#): Quantity of data (in bytes) received from data nodes by this replica.
- [Ndb_api_bytes_sent_count](#): Quantity of data (in bytes) sent to data nodes by this MySQL Server (SQL node).
- [Ndb_api_bytes_sent_count_session](#): Quantity of data (in bytes) sent to data nodes in this client session.
- [Ndb_api_bytes_sent_count_slave](#): Quantity of data (in bytes) sent to data nodes by this replica.
- [Ndb_api_event_bytes_count](#): Number of bytes of events received by this MySQL Server (SQL node).
- [Ndb_api_event_bytes_count_injector](#): Number of bytes of event data received by NDB binary log injector thread.
- [Ndb_api_event_data_count](#): Number of row change events received by this MySQL Server (SQL node).
- [Ndb_api_event_data_count_injector](#): Number of row change events received by NDB binary log injector thread.
- [Ndb_api_event_nondata_count](#): Number of events received, other than row change events, by this MySQL Server (SQL node).

- `Ndb_api_event_nondata_count_injector`: Number of events received, other than row change events, by NDB binary log injector thread.
- `Ndb_api_pk_op_count`: Number of operations based on or using primary keys by this MySQL Server (SQL node).
- `Ndb_api_pk_op_count_session`: Number of operations based on or using primary keys in this client session.
- `Ndb_api_pk_op_count_slave`: Number of operations based on or using primary keys by this replica.
- `Ndb_api_pruned_scan_count`: Number of scans that have been pruned to one partition by this MySQL Server (SQL node).
- `Ndb_api_pruned_scan_count_session`: Number of scans that have been pruned to one partition in this client session.
- `Ndb_api_pruned_scan_count_slave`: Number of scans that have been pruned to one partition by this replica.
- `Ndb_api_range_scan_count`: Number of range scans that have been started by this MySQL Server (SQL node).
- `Ndb_api_range_scan_count_session`: Number of range scans that have been started in this client session.
- `Ndb_api_range_scan_count_slave`: Number of range scans that have been started by this replica.
- `Ndb_api_read_row_count`: Total number of rows that have been read by this MySQL Server (SQL node).
- `Ndb_api_read_row_count_session`: Total number of rows that have been read in this client session.
- `Ndb_api_read_row_count_slave`: Total number of rows that have been read by this replica.
- `Ndb_api_scan_batch_count`: Number of batches of rows received by this MySQL Server (SQL node).
- `Ndb_api_scan_batch_count_session`: Number of batches of rows received in this client session.
- `Ndb_api_scan_batch_count_slave`: Number of batches of rows received by this replica.
- `Ndb_api_table_scan_count`: Number of table scans that have been started, including scans of internal tables, by this MySQL Server (SQL node).
- `Ndb_api_table_scan_count_session`: Number of table scans that have been started, including scans of internal tables, in this client session.
- `Ndb_api_table_scan_count_slave`: Number of table scans that have been started, including scans of internal tables, by this replica.
- `Ndb_api_trans_abort_count`: Number of transactions aborted by this MySQL Server (SQL node).
- `Ndb_api_trans_abort_count_session`: Number of transactions aborted in this client session.
- `Ndb_api_trans_abort_count_slave`: Number of transactions aborted by this replica.
- `Ndb_api_trans_close_count`: Number of transactions aborted (may be greater than sum of `TransCommitCount` and `TransAbortCount`) by this MySQL Server (SQL node).

- [Ndb_api_trans_close_count_session](#): Number of transactions aborted (may be greater than sum of TransCommitCount and TransAbortCount) in this client session.
- [Ndb_api_trans_close_count_slave](#): Number of transactions aborted (may be greater than sum of TransCommitCount and TransAbortCount) by this replica.
- [Ndb_api_trans_commit_count](#): Number of transactions committed by this MySQL Server (SQL node).
- [Ndb_api_trans_commit_count_session](#): Number of transactions committed in this client session.
- [Ndb_api_trans_commit_count_slave](#): Number of transactions committed by this replica.
- [Ndb_api_trans_local_read_row_count](#): Total number of rows that have been read by this MySQL Server (SQL node).
- [Ndb_api_trans_local_read_row_count_session](#): Total number of rows that have been read in this client session.
- [Ndb_api_trans_local_read_row_count_slave](#): Total number of rows that have been read by this replica.
- [Ndb_api_trans_start_count](#): Number of transactions started by this MySQL Server (SQL node).
- [Ndb_api_trans_start_count_session](#): Number of transactions started in this client session.
- [Ndb_api_trans_start_count_slave](#): Number of transactions started by this replica.
- [Ndb_api_uk_op_count](#): Number of operations based on or using unique keys by this MySQL Server (SQL node).
- [Ndb_api_uk_op_count_session](#): Number of operations based on or using unique keys in this client session.
- [Ndb_api_uk_op_count_slave](#): Number of operations based on or using unique keys by this replica.
- [Ndb_api_wait_exec_complete_count](#): Number of times thread has been blocked while waiting for operation execution to complete by this MySQL Server (SQL node).
- [Ndb_api_wait_exec_complete_count_session](#): Number of times thread has been blocked while waiting for operation execution to complete in this client session.
- [Ndb_api_wait_exec_complete_count_slave](#): Number of times thread has been blocked while waiting for operation execution to complete by this replica.
- [Ndb_api_wait_meta_request_count](#): Number of times thread has been blocked waiting for metadata-based signal by this MySQL Server (SQL node).
- [Ndb_api_wait_meta_request_count_session](#): Number of times thread has been blocked waiting for metadata-based signal in this client session.
- [Ndb_api_wait_meta_request_count_slave](#): Number of times thread has been blocked waiting for metadata-based signal by this replica.
- [Ndb_api_wait_nanos_count](#): Total time (in nanoseconds) spent waiting for some type of signal from data nodes by this MySQL Server (SQL node).
- [Ndb_api_wait_nanos_count_session](#): Total time (in nanoseconds) spent waiting for some type of signal from data nodes in this client session.

- `Ndb_api_wait_nanos_count_slave`: Total time (in nanoseconds) spent waiting for some type of signal from data nodes by this replica.
- `Ndb_api_wait_scan_result_count`: Number of times thread has been blocked while waiting for scan-based signal by this MySQL Server (SQL node).
- `Ndb_api_wait_scan_result_count_session`: Number of times thread has been blocked while waiting for scan-based signal in this client session.
- `Ndb_api_wait_scan_result_count_slave`: Number of times thread has been blocked while waiting for scan-based signal by this replica.
- `ndb_autoincrement_prefetch_sz`: NDB auto-increment prefetch size.
- `ndb_cache_check_time`: Number of milliseconds between checks of cluster SQL nodes made by MySQL query cache.
- `ndb_clear_apply_status`: Causes RESET SLAVE/RESET REPLICAS to clear all rows from `ndb_apply_status` table; ON by default.
- `Ndb_cluster_node_id`: Node ID of this server when acting as NDB Cluster SQL node.
- `Ndb_config_from_host`: NDB Cluster management server host name or IP address.
- `Ndb_config_from_port`: Port for connecting to NDB Cluster management server.
- `Ndb_conflict_fn_epoch`: Number of rows that have been found in conflict by `NDB$EPOCH()` conflict detection function.
- `Ndb_conflict_fn_epoch2`: Number of rows that have been found in conflict by `NDB$EPOCH2()` conflict detection function.
- `Ndb_conflict_fn_epoch2_trans`: Number of rows that have been found in conflict by `NDB$EPOCH2_TRANS()` conflict detection function.
- `Ndb_conflict_fn_epoch_trans`: Number of rows that have been found in conflict by `NDB$EPOCH_TRANS()` conflict detection function.
- `Ndb_conflict_fn_max`: Number of times that conflict resolution based on "greater timestamp wins" has been applied when server is part of an NDB Cluster involved in cluster replication.
- `Ndb_conflict_fn_max_del_win`: Number of times that conflict resolution based on outcome of `NDB$MAX_DELETE_WIN()` has been applied.
- `Ndb_conflict_fn_old`: Number of times that "same timestamp wins" conflict resolution has been applied when this server is part of an NDB Cluster involved in cluster replication.
- `Ndb_conflict_last_conflict_epoch`: Most recent NDB epoch on this replica in which some conflict was detected.
- `Ndb_conflict_last_stable_epoch`: Number of rows found to be in conflict by transactional conflict function.
- `Ndb_conflict_reflected_op_discard_count`: Number of reflected operations that were not applied due error during execution.
- `Ndb_conflict_reflected_op_prepare_count`: Number of reflected operations received that have been prepared for execution.

- [Ndb_conflict_refresh_op_count](#): Number of refresh operations that have been prepared.
- [Ndb_conflict_trans_conflict_commit_count](#): Number of epoch transactions committed after requiring transactional conflict handling.
- [Ndb_conflict_trans_detect_iter_count](#): Number of internal iterations required to commit epoch transaction. Should be (slightly) greater than or equal to [Ndb_conflict_trans_conflict_commit_count](#).
- [Ndb_conflict_trans_reject_count](#): Number of transactions rejected after being found in conflict by transactional conflict function.
- [Ndb_conflict_trans_row_conflict_count](#): Number of rows found in conflict by transactional conflict function. Includes any rows included in or dependent on conflicting transactions.
- [Ndb_conflict_trans_row_reject_count](#): Total number of rows realigned after being found in conflict by transactional conflict function. Includes [Ndb_conflict_trans_row_conflict_count](#) and any rows included in or dependent on conflicting transactions.
- [ndb_deferred_constraints](#): Specifies that constraint checks should be deferred (where these are supported). Not normally needed or used; for testing purposes only.
- [ndb_distribution](#): Default distribution for new tables in NDBCLUSTER (KEYHASH or LINHASH, default is KEYHASH).
- [Ndb_epoch_delete_delete_count](#): Number of delete-delete conflicts detected (delete operation is applied, but row does not exist).
- [ndb_eventbuffer_free_percent](#): Percentage of free memory that should be available in event buffer before resumption of buffering, after reaching limit set by [ndb_eventbuffer_max_alloc](#).
- [ndb_eventbuffer_max_alloc](#): Maximum memory that can be allocated for buffering events by NDB API. Defaults to 0 (no limit).
- [Ndb_execute_count](#): Number of round trips to NDB kernel made by operations.
- [ndb_extra_logging](#): Controls logging of NDB Cluster schema, connection, and data distribution events in MySQL error log.
- [ndb_force_send](#): Forces sending of buffers to NDB immediately, without waiting for other threads.
- [ndb_index_stat_cache_entries](#): Granularity of statistics by determining number of starting and ending keys.
- [ndb_index_stat_enable](#): Use NDB index statistics in query optimization.
- [ndb_index_stat_option](#): Comma-separated list of tunable options for NDB index statistics; list should contain no spaces.
- [ndb_index_stat_update_freq](#): How often to query data nodes rather than statistics cache.
- [ndb_join_pushdown](#): Enables pushing down of joins to data nodes.
- [Ndb_last_commit_epoch_server](#): Epoch most recently committed by NDB.
- [Ndb_last_commit_epoch_session](#): Epoch most recently committed by this NDB client.
- [ndb_log_apply_status](#): Whether or not MySQL server acting as replica logs `mysql.ndb_apply_status` updates received from its immediate source in its own binary log, using its own server ID.

- `ndb_log_bin`: Write updates to NDB tables in binary log. Effective only if binary logging is enabled with `--log-bin`.
- `ndb_log_binlog_index`: Insert mapping between epochs and binary log positions into `ndb_binlog_index` table. Defaults to ON. Effective only if binary logging is enabled.
- `ndb_log_empty_epochs`: When enabled, epochs in which there were no changes are written to `ndb_apply_status` and `ndb_binlog_index` tables, even when `log_slave_updates` is enabled.
- `ndb_log_empty_update`: When enabled, updates which produce no changes are written to `ndb_apply_status` and `ndb_binlog_index` tables, even when `log_slave_updates` is enabled.
- `ndb_log_exclusive_reads`: Log primary key reads with exclusive locks; allow conflict resolution based on read conflicts.
- `ndb_log_orig`: Whether id and epoch of originating server are recorded in `mysql.ndb_binlog_index` table. Set using `--ndb-log-orig` option when starting `mysqld`.
- `ndb_log_transaction_id`: Whether NDB transaction IDs are written into binary log (Read-only).
- `ndb-log-update-minimal`: Log updates in minimal format.
- `ndb-log-updated-only`: Log complete rows (ON) or updates only (OFF).
- `Ndb_number_of_data_nodes`: Number of data nodes in this NDB cluster; set only if server participates in cluster.
- `ndb-optimization-delay`: Number of milliseconds to wait between processing sets of rows by `OPTIMIZE TABLE` on NDB tables.
- `ndb_optimized_node_selection`: Determines how SQL node chooses cluster data node to use as transaction coordinator.
- `Ndb_pruned_scan_count`: Number of scans executed by NDB since cluster was last started where partition pruning could be used.
- `Ndb_pushed_queries_defined`: Number of joins that API nodes have attempted to push down to data nodes.
- `Ndb_pushed_queries_dropped`: Number of joins that API nodes have tried to push down, but failed.
- `Ndb_pushed_queries_executed`: Number of joins successfully pushed down and executed on data nodes.
- `Ndb_pushed_reads`: Number of reads executed on data nodes by pushed-down joins.
- `ndb_recv_thread_activation_threshold`: Activation threshold when receive thread takes over polling of cluster connection (measured in concurrently active threads).
- `ndb_recv_thread_cpu_mask`: CPU mask for locking receiver threads to specific CPUs; specified as hexadecimal. See documentation for details.
- `ndb_report_thresh_binlog_epoch_slip`: NDB 7.5.4 and later: Threshold for number of epochs completely buffered, but not yet consumed by binlog injector thread which when exceeded generates `BUFFERED_EPOCHS_OVER_THRESHOLD` event buffer status message; prior to NDB 7.5.4: Threshold for number of epochs to lag behind before reporting binary log status.
- `ndb_report_thresh_binlog_mem_usage`: Threshold for percentage of free memory remaining before reporting binary log status.

- `Ndb_scan_count`: Total number of scans executed by NDB since cluster was last started.
- `ndb_show_foreign_key_mock_tables`: Show mock tables used to support `foreign_key_checks=0`.
- `ndb_slave_conflict_role`: Role for replica to play in conflict detection and resolution. Value is one of PRIMARY, SECONDARY, PASS, or NONE (default). Can be changed only when replication SQL thread is stopped. See documentation for further information.
- `Ndb_slave_max_replicated_epoch`: Most recently committed NDB epoch on this replica. When this value is greater than or equal to `Ndb_conflict_last_conflict_epoch`, no conflicts have yet been detected.
- `ndb_table_no_logging`: NDB tables created when this setting is enabled are not checkpointed to disk (although table schema files are created). Setting in effect when table is created with or altered to use NDBCLUSTER persists for table's lifetime.
- `ndb_table_temporary`: NDB tables are not persistent on disk: no schema files are created and tables are not logged.
- `ndb_use_copying_alter_table`: Use copying ALTER TABLE operations in NDB Cluster.
- `ndb_use_exact_count`: Use exact row count when planning queries.
- `ndb_use_transactions`: Forces NDB to use a count of records during SELECT COUNT(*) query planning to speed up this type of query.
- `ndb_version`: Shows build and NDB engine version as an integer.
- `ndb_version_string`: Shows build information including NDB engine version in `ndb-x.y.z` format.
- `ndbcluster`: Enable NDB Cluster (if this version of MySQL supports it). Disabled by `--skip-ndbcluster`.
- `ndbinfo_database`: Name used for NDB information database; read only.
- `ndbinfo_max_bytes`: Used for debugging only.
- `ndbinfo_max_rows`: Used for debugging only.
- `ndbinfo_offline`: Put `ndbinfo` database into offline mode, in which no rows are returned from tables or views.
- `ndbinfo_show_hidden`: Whether to show `ndbinfo` internal base tables in `mysql` client; default is OFF.
- `ndbinfo_table_prefix`: Prefix to use for naming `ndbinfo` internal base tables; read only.
- `ndbinfo_version`: `ndbinfo` engine version; read only.
- `server_id_bits`: Number of least significant bits in `server_id` actually used for identifying server, permitting NDB API applications to store application data in most significant bits. `server_id` must be less than 2 to power of this value.
- `skip-ndbcluster`: Disable NDB Cluster storage engine.
- `slave_allow_batching`: Turns update batching on and off for replica.
- `transaction_allow_batching`: Allows batching of statements within one transaction. Disable AUTOCOMMIT to use.

Index

A

- abort backup command, 137
 - backupid option, 137
- add hosts command, 78
- add package command, 86, 89
 - basedir option, 87
- add process command, 128
- agent
 - backing up and restoring, 60, 140
 - configuration, 19
 - defined, 2, 2
 - distribution, 6, 6
 - distribution layout, 7
 - installing, 6
 - starting, 31
 - starting (Linux), 32
 - starting (Windows), 33
 - stopping (Linux), 33
 - stopping (Windows), 35
- agent-uuid option
 - mcmd, 25
- architecture, 1, 2
- attributes
 - case-sensitivity, 67
 - summary table, 149

B

- background option
 - limitations, 145
- backgrounding of commands, 145
- backup
 - commands, 137
- backup agents command, 140
- backup cluster command, 137
 - snapshotend option, 137
 - snapshotstart option, 137
 - waitcompleted option, 137
 - waitstarted option, 137
- backup images, 138
- backup option (show status command), 98
- backup status, 98
- backupid option (abort backup), 137
- backupid option (restore cluster), 139
- backups
 - creating, 60, 137, 140
 - removing, 138
- basedir option
 - mcmd, 25
- basedir option (add package command), 87
- bootstrap option

mcmd, 25

C

- change log-level command, 80
- change process command, 131
 - limitations, 143
- changing data node processes, 131
- client
 - commands in, 66
 - defined, 2, 3
 - executing scripts with, 73
 - importing, 39
 - mysql client commands in, 73
 - starting, 35
- client commands, 66
 - abort backup, 137
 - add hosts, 78
 - add package, 86, 89
 - add process, 128
 - backup, 137
 - backup agents, 140
 - backup cluster, 137
 - case-sensitivity in, 67
 - change log-level, 79
 - change process, 131
 - cluster, 90
 - collect logs, 81
 - command-specific, 72
 - configuration, 103
 - create cluster, 90
 - create site, 81
 - delete cluster, 94
 - delete package, 88
 - delete site, 82
 - get, 106
 - help, 78
 - identifiers in, 67
 - import cluster, 141
 - import config, 141
 - importing clusters, 141
 - information, 71
 - list backups, 138
 - list clusters, 95
 - list hosts, 83
 - list nextnodeids, 95
 - list processes, 134
 - list sites, 83
 - online help, 71
 - options, 66, 69, 70
 - package, 86
 - process, 128
 - remove hosts, 79
 - remove process, 135

- reset, 115
- restart cluster, 95
- restore, 137
- restore cluster, 138
- results returned by, 70
- rotate log, 80
- set, 122
- show settings, 84
- show status, 96
- show warning, 85
- site, 78
- start cluster, 100
- start process, 134
- status, 97
- stop agents, 84
- stop cluster, 101
- stop process, 135
- syntax, 66
- upgrade cluster, 102
- version, 85
- cluster
 - defined, 1
- cluster backups
 - aborting, 137
 - creating, 56
 - listing, 138
 - restoring, 56
 - restoring from, 138
- cluster option (show status command), 96
- cluster processes
 - adding, 128
- cluster status, 96
- clusters
 - creating, 38, 38, 90
 - creating for import, 92
 - importing, 141, 141
 - listing, 95
 - removing, 94
 - restarting, 95
 - starting, 100
 - stopping, 101
 - upgrading, 102
- collect logs command, 81
- command status, 97
- common terms, 1
- concurrent client sessions unsupported, 143
- configuration
 - derivation of attributes, 104
- configuration attributes, 104
 - defined, 2
 - for TCP connections, 126
 - getting, 106
 - how determined by MySQL Cluster Manager, 105
 - levels applying, 104

- mandatory, 105
- read-only, 105
- resetting, 115
- setting, 122

- configuration commands, 103
- configuration data
 - importing, 141
- configuration file, 19
- configuration parameters (see [configuration attributes](#))
- configuration variables (see [configuration attributes](#))
- connecting to agent
 - with mcm client, 36
 - with mysql client, 36
- create cluster command, 90
 - import option, 92
 - assignment of node IDs in, 92
- create site command, 81
 - hosts option, 82

D

- daemon option
 - mcmd, 25
- defaults-file option
 - mcmd, 25
- delete cluster command, 94
- delete package command, 88
- delete site command, 82
- deployment (example), 3
- disable-indexes option (restore cluster), 139
- disable-metadata option (restore cluster), 139

E

- epoch option (restore cluster), 139
- event-threads option
 - mcmd, 25
- exclude-databases option (restore cluster), 139
- exclude-intermediate-sql-tables option (restore cluster), 139
- exclude-missing-columns option (restore cluster), 139
- exclude-missing-tables option (restore cluster), 139
- exclude-tables option (restore cluster), 139

G

- get command, 106
 - filtering output, 111
 - include-defaults option, 109
 - with multiple attributes, 112

H

- help command, 78
- help for commands, 72
- help option, 72
 - mcmd, 26

- help-all option
 - mcmd, 26
- help-manager option
 - mcmd, 27
- hosts
 - defined, 1
- hosts option (create site), 82

I

- identifiers, 67
 - case-sensitivity, 67
 - spaces in, 68
- import cluster command, 141
- import config command, 141
- importing clusters
 - creating for import, 92
 - example, 41
 - limitations, 145
- include-databases option (restore cluster), 140
- include-defaults option (get command), 109
- include-tables option (restore cluster), 140
- initial option (start cluster), 101
- installation, 5
- IPv6
 - and hostnames (Windows), 143

K

- keepalive option
 - mcmd, 27

L

- license keys
 - and operating platform, 143
- license keys (obsolete), 6
- limitations, 143
 - background option, 145
 - backup, 145, 145
 - change process command, 143
 - client, 144, 144, 144, 144
 - cluster imports, 145
 - concurrent usage, 143
 - IPv6, 143
 - license keys, 143
 - MySQL server variables, 144, 144
 - MySQL user variables, 144
 - operating systems, 143
 - relating to MySQL Server, 144
 - replication, 145
 - restarts, 145
 - syntax, 145
 - Windows, 143
- list backups command, 138
- list clusters command, 95

- list commands command, 71
- list hosts command, 83
- list nextnodeids command, 95
- list processes command, 134
- list sites command, 83
- log-backtrace-on-crash option
 - mcmd, 27
- log-file option
 - mcmd, 27
- log-level option
 - mcmd, 28
- log-use-syslog option
 - mcmd, 29
- logging
 - configuring, 80
- logs
 - obtaining, 81
 - rotating, 80
- lossy-conversions option (restore cluster), 140

M

- management site
 - defined, 1
- management sites
 - adding hosts, 78
 - creating, 81
 - deleting, 82
 - listing, 83
 - listing hosts, 83
 - removing hosts, 79
- manager-directory option
 - mcmd, 29
- manager-password option
 - mcmd, 29
- manager-port option
 - mcmd, 29
- manager-username option
 - mcmd, 30
- max-open-files option
 - mcmd, 30
- mcm client
 - and mysql client, 35
- mcmd, 32
 - agent-uuid option, 25
 - basedir option, 25
 - bootstrap option, 25
 - daemon option, 25
 - defaults-file option, 25
 - event-threads option, 25
 - help option, 26
 - help-all option, 26
 - help-manager option, 27
 - keepalive option, 27

- log-backtrace-on-crash option, 27
- log-file option, 27
- log-level option, 28
- log-use-syslog option, 29
- manager-directory option, 29
- manager-password option, 29
- manager-port option, 29
- manager-username option, 30
- max-open-files option, 30
- pid-file option, 30
- plugin-dir option, 30
- plugins option, 31
- verbose-shutdown option, 31
- version option, 31
- xcom-port option, 31

mcmd.exe, 33

multiple client sessions, 143

mysql client commands, 73

mysql-cluster-manager (OBSOLETE, see mcmd), 32

mysqld options, 144

N

ndb_mgm (MySQL Cluster command-line client)

- using with MySQL Cluster Manager, 128

no-binlog option (restore cluster), 140

no-restore-disk-objects option (restore cluster), 140

node IDs

- and create cluster command, 92
- listing, 95

O

obtaining MySQL Cluster Manager, 5

operation option (show status command), 97

P

packages

- defined, 2
- listing, 86, 89
- registering, 86
- removing, 88

parallelism option (restore cluster), 140

pid-file option

- mcmd, 30

plugin-dir option

- mcmd, 30

plugins option

- mcmd, 31

privilege-tables option (restore cluster), 140

process option (show status command), 98

process status, 98

processes

- changing, 131
- commands, 128

- defined, 1
- listing, 134
- removing, 135
- starting, 134
- status, 98
- stopping, 135

progress-frequency option (restore cluster), 140

R

ReceiveBufferMemory, 121, 126

remove hosts command, 79

remove process command, 135

replication, 145

- setup, 62

reset command, 115

- and attribute name, 116
- and TCP connections, 121
- order of operands, 119
- process level, 116
- scope, 116

restart cluster command, 95

restore (from backup)

- commands, 137

restore cluster command, 138

- backupid option, 139
- disable-indexes option, 139
- disable-metadata option, 139
- epoch option, 139
- exclude-databases option, 139
- exclude-intermediate-sql-tables option, 139
- exclude-missing-columns option, 139
- exclude-missing-tables option, 139
- exclude-tables option, 139
- include-databases option, 140
- include-tables option, 140
- lossy-conversions option, 140
- no-binlog option, 140
- no-restore-disk-objects option, 140
- parallelism option, 140
- privilege-tables option, 140
- progress-frequency option, 140
- rewrite-database option, 140
- skip-broken-objects option, 140
- skip-nodeid option, 140
- skip-table-check option, 140
- skip-unknown-objects option, 140

rewrite-database option (restore cluster), 140

rolling restarts, 145

rotate log command, 80

S

scripts (MySQL Cluster Manager client), 73

SendBufferMemory, 121, 126

set command, 122
 and TCP connection attributes, 126
 instance level, 123
 paths used with (Windows), 125
 scope, 122, 123
 undoing effects of, 117
 verifying effects, 123
 with multiple attributes, 125
 with multiple processes, 124

show settings command, 84

show status command, 96
 backup option, 98
 cluster option, 96
 operation option, 97
 process option, 98

skip-broken-objects option (restore cluster), 140

skip-nodeid option (restore cluster), 140

skip-table-check option (restore cluster), 140

skip-unknown-objects option (restore cluster), 140

snapshotend option (backup cluster), 137

snapshotstart option (backup cluster), 137

start cluster command, 100
 initial option, 101

start process command, 134

starting and stopping nodes
 and ndb_mgm (MySQL Cluster command-line client),
 128

stop agents command, 84

stop cluster command, 101

stop process command, 135

syntax issues, 145

mcmd, 31

T

terminology, 1

U

upgrade cluster command, 102

upgrades

 MySQL Cluster, 145

V

verbose-shutdown option

 mcmd, 31

version command, 85

version option

 mcmd, 31

W

waitcompleted option (backup cluster), 137

waitstarted option (backup cluster), 137

X

xcom-port option

