Oracle® Enterprise Manager

Command Line Interface

10*g* Release 4 (10.2.0.4)

B40004-02

October 2007

ORACLE®

Oracle Enterprise Manager Command Line Interface, 10*g* Release 4 (10.2.0.4)

B40004-02

Contents

3 Error Code Reference

Index

Preface

This manual covers installation and error codes for the Enterprise Manager Command Line Interface (EM CLI). A complete verb reference, which duplicates the command line help, is also included.

Note that more recent versions of this and other Enterprise Manager books are available on the Oracle Technology Network:

http://www.oracle.com/technology/documentation/oem.html

Audience

This guide is written for administrators who want to access Enterprise Manager console functions directly from scripts or interactively from an OS shell. You should already be familiar with Enterprise Manager administrative tasks you want to perform.

You should also be familiar with the operation of your specific UNIX or Windows system. Refer to your platform-specific documentation if necessary.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at:

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documents

For more information, see the following manuals in the Oracle Enterprise Manager 10*g* Release 2 documentation set:

- *Oracle Enterprise Manager Administrator's Guide*
- *Oracle Enterprise Manager Concepts*
- *Oracle Enterprise Manager Grid Control Quick Installation Guide*
- *Oracle Enterprise Manager Grid Control Installation and Basic Configuration*
- *Oracle Enterprise Manager Configuration for Oracle Collaboration Suite*
- *Oracle Enterprise Manager Policy Reference Manual*
- *Oracle Enterprise Manager Metric Reference Manual*
- *Oracle Enterprise Manager Extensibility*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Command Line Interface

This chapter discusses the following Enterprise Manager Command Line Interface (EM CLI) topics:

- Overview
- How the EM CLI Works
- Preliminary Advisory Information
- EM CLI Quick Start
- Security and Authentication
- EM CLI General Behavior
- Using EM CLI

1.1 Overview

The Enterprise Manager Command Line Interface (EM CLI) allows you to access Enterprise Manager Grid Control functionality from text-based consoles (shells and command windows) for a variety of operating systems. This capability enables administrators to call Enterprise Manager functionality using custom scripts, such as SQL*Plus, OS shell, Perl, or Tcl, thus easily integrating Enterprise Manager functionality with a company's business process.

Using EM CLI, you can perform Enterprise Manager Grid Control console-based operations, like monitoring/managing targets, jobs, groups, blackouts, notifications, and alerts. EM CLI is intended for use by enterprise or system administrators writing scripts such as shell/batch files, Perl, Tcl, or PHP that provide workflow in the customer's business process. EM CLI commands can also be used interactively from an operating system console.

EM CLI is fully integrated with Enterprise Manager's security and user administration functions, thus allowing administrators to carry out operations using EM CLI with the same security and confidentiality as the Enterprise Manager Grid Control console. For example, the EM CLI user can only see and operate on targets for which they are authorized.

Examples of EM CLI usage are as follows:

- Enterprise Manager Integration with third-party or custom software through scripting. Actions (such as add/delete target, submit/delete jobs, create/delete user) that are part of a customer's business model can be performed through scripting.
- Every day, send an e-mail list of backup jobs that were still running after 6 AM.

- Every week, write pertinent information about failed Enterprise Manager jobs to a file and then purge the Enterprise Manager job history.

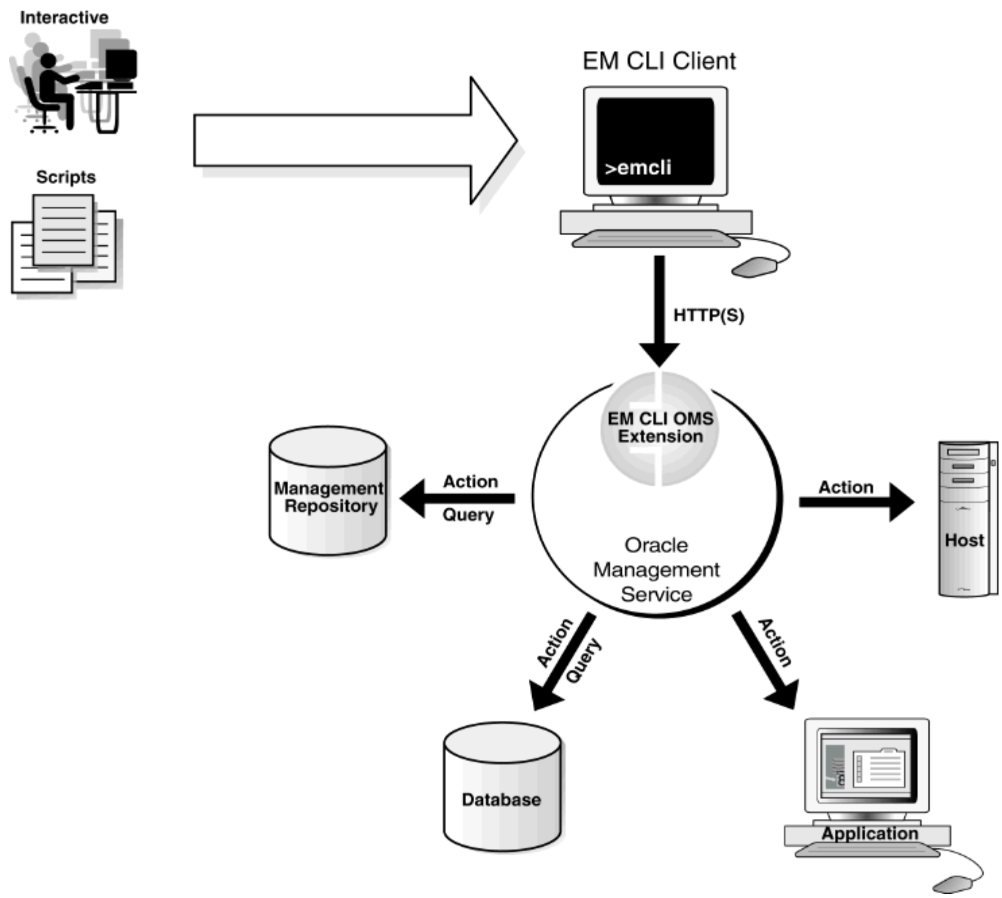
1.2 How the EM CLI Works

The EM CLI Client is a Java application that accepts a command as input. The EM CLI Client then uses the input command to identify a Verb to execute the command. A Verb is a Java plug-in extension to the EM CLI Client. A Verb services the command with its specific options and posts the results to the standard output stream. Any errors are posted to the error output stream. The Verb also returns an integer exit value that the EM CLI Client sets as the exit value of the command in the Client's calling environment (the operating system console).

A Verb can perform its operations locally, but most of the verbs included with the EM CLI are covered by the remote Verb in the EM CLI Client. The remote Verb contacts the EM CLI OMS Extension in the Enterprise Manager OMS Console via HTTP/HTTPS and sends the command line through HTTP to the OMS for processing. The EM CLI OMS Extension is essentially a standard Enterprise Manager console page, and is installed in the OMS just as any other standard console page. As with the EMCLI Client, the EM CLI OMS Extension uses the input command to identify a Verb to execute the command. The Verb can access the Management Repository or Management Agents via OMS services as necessary in processing the command.

The remote Verb logs on to the OMS and establishes a session automatically, as necessary, to access the OMS-Side Controller. The remote Verb impersonates the Enterprise Manager user that invoked the command from the Client. The Enterprise Manager user credentials are established local to the EM CLI Client during a one-time, interactive exchange when the Enterprise Manager administrator uses the EM CLI `setup` Verb. Figure 1–1 shows the high-level architecture of EM CLI.

Figure 1–1 EM CLI Architecture



For more information about any of these functional areas, see *Oracle Enterprise Manager Concepts*.

1.3 Preliminary Advisory Information

- EM CLI setup Verb does not work if you have executed a UNIX "su" or "msu" command to operate as another user in the current window. To run the "setup" Verb as another user, open a new terminal window and run the "setup" Verb as the new user in that window.
- EM CLI does not allow OS Script jobs to be run against database targets. The Enterprise Manager Grid Control console, however, does allow this.
- EM CLI has only been certified for submitting OS Script and SQL Script jobs.
- The current version of EM CLI (10.2.04) does not work with Oracle Management Services (OMS) with single sign-on (SSO) enabled.

1.4 EM CLI Quick Start

Setting up and running EM CLI is simple. EM CLI consists of two components used to access the Enterprise Manager framework functionality:

- EM CLI Client
- EM CLI Oracle Management Service Extension

You can install the EM CLI Client on any machine within your managed network. The EM CLI Client is a command-line program (Java-based) that sends EM CLI Verbs to a specific Oracle Management Service (OMS). In some respects, the EM CLI Client functions as a command-line equivalent of an Enterprise Manager Grid Control console. The EM CLI Oracle Management Service Extension is automatically installed with the OMS and serves as the communication conduit between the EM CLI Client and the OMS.

Requirements

Before installing EM CLI, you will need the following:

- Enterprise Manager 10g 10.2.0.4 Grid Control framework
- Java version 1.4.2 or greater
- Workstation running Solaris, Linux, HPUX, Tru64, AIX, or Windows with NTFS.

1.4.1 Installation and Setup

As mentioned above, the EM CLI OMS Extension is automatically installed with the OMS. You must install and set up the client portion. The following instructions cover installation and setup procedures for the EM CLI Client.

1.4.1.1 Installing the EM CLI Client

1. Obtain the EM CLI Client kit (emclikit.jar).

   The EM CLI client kit is downloadable from any 10.2 Grid Control installation at the following location:

   ```
   HTTP(S)://host:port/em/console/emcli/download
   ```

   The `emclikit.jar` file is physically located in the `$ORACLE_HOME/sysman/jlib directory` of the 10.2.0.4 Grid Control OMS home.

2. Set your `JAVA_HOME` environment variable and ensure that it is part of your `PATH`. Make sure that this variable is set to the home of a JDK 1.4.2 or greater. For example:

   ```
   setenv JAVA_HOME /usr/local/packages/j2sdk1.4.2_02

   setenv PATH $JAVA_HOME/bin:$PATH
   ```

3. Install the EM CLI Client. You can install the client in any directory either on the same machine as the EM CLI Management Services or on any machine in your network (download the `emclikit.jar` file to that machine). Run the following command:

   java -jar emclikit.jar client -install_dir=<emcli client dir>

After you have installed the EM CLI Client, you are ready to begin client setup.

1.4.1.2 Setting Up the EM CLI Client

After the EM CLI Client is installed, you are ready to begin using EM CLI. At this point, you can run the EM CLI Client out of the "install_dir" location, or alternatively, you can add it to your `PATH`.

Immediately after installation, only basic operational Verbs are installed:

- **argfile** — Execute an EM CLI Verb where the Verb and any arguments are contained in a file.

- **help** — Access command-line help for EM CLI Verbs.
- **setup** — Configure EM CLI to work with a specific Enterprise Manager Management Service (OMS).
- **sync** — Synchronize the EM CLI Client with an OMS.
- **add_mp_to_mpa** — Create (or adds to) the Management Plug-in Archive. The Management Plug-in Archive is used for adding new target types to Enterprise Manager.
- **add_group_to_mpa** — Adds a Management Plug-in group to a Management Plug-in Archive.

You must run "setup" to connect the EM CLI Client to the OMS running the EM CLI Management Services. For information on how to use the `setup` Verb, see the command-line help by entering the following command:

> emcli help setup

Example 1–1 Setting Up the EM CLI Client

```
emcli setup -url=http://myworkstation.us.oracle.com:em_port/em -username=em_user
```

When the EM CLI Client connects with the EM CLI Management Services, you are prompted at the command line to enter the user password. Running the `setup` Verb installs all available Verb-associated command-line help from the EM CLI Management Service. You must run `setup` each time you want to connect to a different OMS.

After running the `setup` Verb, you are ready to begin using EM CLI.

1.4.1.3 EM CLI Log Files

EM CLI creates log files to record informational and error messages generated during operation. Not all of the logs in the following examples will necessarily be present. Logs are created as needed and are append-based—they are preserved between invocations of EM CLI. You can safely delete log files at any time without affecting EM CLI operation.

The logs contain stack traces, which may not be useful for the casual user, but may benefit users with a high level of system knowledge. The following examples show possible log file locations:

```
CONFIG_DIR/.emcli.log
CONFIG_DIR/.emcli.log.1
```

`CONFIG_DIR` refers to the directory specified by the `-dir` option in the latest running of the `setup` Verb (with an appended `.emcli` subdirectory). The current `CONFIG_DIR` directory can be identified by executing the `setup` Verb with no options to display the setup summary. Log files are limited to a maximum of 0.5 MB. EM CLI alternates between the two log files — as each file reaches the 0.5 MB limit, EM CLI begins writing to the other file, overwriting the oldest log file after `emcli.log.1` has been filled for the first time.

The following examples show possible log file locations:

Example 1–2 No configuration directory is specified with the setup Verb (Default location)

```
user.home/.emcli/.emcli.log
user.home/.emcli/.emcli.log.1
```

If no configuration directory is specified when the `setup` Verb is run (`-dir` option is omitted), EM CLI assumes the `.emcli` configuration directory is located within the user's local home directory. The log files are placed at the root level of the `.emcli` directory. The `.emcli` directory must be local (not mounted remotely).

Example 1–3 Local configuration directory is specified with the setup Verb (-dir=<local directory>

```
local.dir/.emcli/.emcli.log
local.dir/.emcli/.emcli.log.1
```

In this example, the configuration directory is specified using the `-dir` option when the `setup` Verb is run. This allows you to specify a local configuration directory if the user home directory is mounted remotely (through NFS, for example).

1.4.1.4 Configuring an HTTP Proxy Environment

If you are planning to use EM CLI through an HTTP proxy server, you need to set an additional environment variable (`EMCLI_OPTS`) that supplies EM CLI with the requisite proxy host and port information. The following examples illustrate setting the `EMCLI_OPTS` environment variable for both Windows and UNIX operating systems.

Example 1–4 Setting EMCLI_OPTS in a Microsoft Windows Environment

```
>set EMCLI_OPTS=-Dhttp.proxyHost=<proxy host> -Dhttp.proxyPort=<proxy port>
```

Example 1–5 Setting EMCLI_OPTS in a UNIX Environment (TCSH)

```
>setenv EMCLI_OPTS "-Dhttp.proxyHost=<proxy host> -Dhttp.proxyPort=<proxy port>"
```

1.4.2 EM CLI Command-line Help

The EM CLI incorporates a comprehensive command-line help system that provides various levels of assistance. Available from any EM CLI Client installation, the help system provides a listing of all available verbs, descriptive overviews for each verb, syntax, as well as usage examples. The command-line help is the definitive EM CLI information source.

To access command-line help, type the following:

>emcli help
Provides an overview of all available verbs.

OR

>emcli help <verb>
Provides a detailed description of the Verb, Verb arguments and options, and usage examples.

1.5 Security and Authentication

Each operating system user must execute a one-time EM CLI initialization that locally defines the location of the Oracle Management Services and the Enterprise Manager credentials to be used whenever this user invokes EM CLI.

Example 1–6 CLI-Enterprise Manager Authentication

```
>emcli setup -url="http[s]://host:port/em/" -username="<username>"  [-trustall]
[-novalidate]

>please enter password:
```

Note: You can find out the OMS connection information from any EM CLI Client by issuing the `setup` Verb without any options. For example:

>emcli setup

Oracle Enterprise Manager 10g Release 10.2.0.4

CONFIG DIRECTORY: `/home/emcli_install_dir/.emcli`

OMS : http://my_machine.my_co.com:port/em/

EM USER : `username`

TRUST ALL : false

1.5.1 HTTPS Trusted Certificate Management

For authenticating an OMS during the SSL server authentication phase of an HTTPS connection handshake, EM CLI searches for trusted certificates in the following key stores:

```
CONFIG_DIR/.emcli/.localkeystore
user.home/.emcli/.keystore
JRE_HOME/lib/security/cacerts
```

`CONFIG_DIR` is the directory specified by the `-dir` option in the latest running of the `setup` Verb (with an appended `.emcli` subdirectory). See "EM CLI Log Files" on page 1-5 for more information about the `CONFIG_DIR` parameter.

`JRE_HOME` in a JDK installation is typically `JAVA_HOME/jre`.

The JDK `keytool` command can manage the key stores. For more information about this tool, see the security documentation for your Java VM installation, or at the time of this writing:

`http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html`

Not all of the key stores in the list above will necessarily be present.

1.5.2 Secure Clients

You can provide credentials to EM CLI in one of two ways:

- Provide credentials at the time of use.
- Make credentials persistent on the host machine where the EM CLI client is running, as might be the case when executing EM CLI verbs from a shell script.

Caution: You should only make credentials persistent on hosts when the host is a secure client, since the only protection available for credentials is the file-system security of the OS.

Oracle also recommends not using persistent credentials if the EM CLI user's home directory is mounted over NFS or any other insecure file system.

1.6 EM CLI General Behavior

For easy parsing of Verb output by scripts, a `-script` argument is available for all verbs that generate output data. If the `-script` argument is used, all output columns become tab-separated (with non-null values), and all rows become newline-separated. You can override the default column and row separators by using the `-format` argument in place of `-script`.

```
[ -script | -format="name:<format type>;column_
separator:<separator_text>;row_separator:<separator_text>"]
```

Supported `-format` arguments are shown in Table 1–1.

Table 1–1 Supported "-format" Arguments

Argument	Explanation
`-format="name:pretty"`	Pretty-print the output. This is the default when both `-script` and `-format` are not specified.
`-format="name:script"`	Identical to just specifying `-script`. Columns are tab-separated, and rows are newline- separated.
`-format="name:script;column_separator:<column_sep_string>"`	Causes the Verb output to be column-separated by `<column_sep_string>`. Rows are separated by the newline character.
`-format="name:script;row_separator:<row_sep_string>"`	Causes the Verb output to be row-separated by `<row_sep_string>`. Columns are separated by the tab character.
`-format="name:script;column_separator:<column_sep_string>;row_separator:<row_sep_string>"`	Causes the Verb output to be column-separated by `<column_sep_string>` and row-separated by `<row_sep_string>`.
`-format="name:csv"`	Produces a table with the columns separated by commas and the rows by newlines.

- `-script` is equivalent to `-format="name:script;column_separator:\u0009;row_separator:\u000A"`
- The values for column and row separator are given as one or more character strings. Any of the characters can be represented by the unicode sequence `\uXXXX` (where `X` is a hex value).

NOTE: The ASCII character set is represented by `\u00XX` where `XX` can range from 00 to 7F. For example, the tab character is represented by `\u0009` and the newline character is represented by `\u000A`.

- The `pretty` format type has no attributes.
- In `script` mode, any Verb output cells that contain the separator strings are substituted with the unicode values for these strings, so that the output does not break any scripts required to parse the output.
- `script` is the only format type for which separators can be specified.
- Separators need not be single characters, and can be specified using both regular characters interspersed with unicode sequences as shown in the following example:

Example 1–7 Complex Separator

Separator Specification: `xxx\u0009xxx\u0009`

This separator appears as `xxx` followed by a tab, followed by `xxx` followed by another tab.

1.7 Using EM CLI

As mentioned previously, the EM CLI provides programmatic access to the functions of the Enterprise Manager Grid Control framework. You can access a subset of framework functions from the Enterprise Manager console through the command line using the EM CLI Verbs. See Chapter 2, "Verb Reference" for more information on available verbs.

2

Verb Reference

This chapter provides a complete listing of all EM CLI Verbs in alphabetical order. Complete syntax and usage information is also available for each Verb through EM CLI's command line help system.

Verb List

The following list provides the names of all Verbs and their associated pages where you can find the definition, format, options, and examples for each Verb.

add_beacon

Adds a beacon to the monitoring set of beacons. All enabled tests are pushed to the beacon.

Format

```
add_beacon
      -name=target name
      -type=target type
      -bcnName=beacon name
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **bcnName**

  Beacon name.

Examples

The following example adds MyBeacon to MyTarget service target of type `generic_service`.

```
emcli add_beacon -name='MyTarget' -type='generic_service'
      -bcnName='MyBeacon'
```

add_group_to_mpa

Adds a Management Plug-in (MP) group to a Management Plug-in Archive (MPA). If the MPA file does not exist, it is created.

Format

```
add_group_to_mpa
     -mpa="mpa"
     -name="group name"
     -member="mpname:mpversion"...
     [-desc="description"]

[ ] denotes that the parameter is optional
```

Options

- **mpa**

  Name of the MPA where the resulting MP is placed. The MPA file name could be an existing MPA file or a new file. You can only use this option once in the command.

- **name**

  To be provided.

- **member**

  To be provided.

- desc

  To be provided.

Examples

The following example adds a group that contains a single Management Plug-in.

```
emcli add_group_to_mpa
       -mpa=\"MyMPA.jar\"
       -name=\"MyGroup\"
       -desc=\"MyGroup is described by this text.\"
       -member=\"an_mp:1.1\"
```

The following example adds a group that contains multiple Management Plug-ins. On deployment, `an_mp` is deployed before `another_mp`. The newest imported version of another_mp is used.

```
emcli add_group_to_mpa
      -mpa=\"MyMPA.jar\"
      -name=\"AnotherGroup\"
      -desc=\"AnotherGroup is described by this text.\"
      -member=\"an_mp:1.1\"
      -member=\"another_mp:any\"
```

add_mp_to_mpa

Adds a Management Plug-in (MP) to a Management Plug-in Archive (MPA). If the MPA file does not exist, it is created.

Format

```
add_mp_to_mpa
      -mpa="mpa"
      -mp_version="mp_version"
      -ttd="ttd"
      -dc="dc"
      [-oms_version="oms_version"]
      [-agent_version="agent_version"]
      [-file="file_type:file_path"]...
      [-func_desc="func_desc"]
      [-req_desc="req_desc"]

[ ] denotes that the parameter is optional
```

Options

- **mpa**

  Name of the MPA where the resulting MP is placed. The MPA file name could be an existing MPA file or a new file. You can only use this option once in the command.

- **mp_version**

  Version of the MP being added to the MPA. This version indicates the version of the files that comprise the MP, and is independent of the metadata version in the target type definition file. This version, along with the MP name (the target type as parsed from the target type definition file), indicate a unique MP.

- **ttd**

  Path of the target-type definition file. This file specifies the metadata definition of the target type and the metrics for this target type. You can only use this option once in the command.

- **dc**

  Path of the default collection file. This file specifies the scheduled collection of metrics for targets with this target type. You can only use this option once in the command.

- **oms_version**

  Minimum OMS version compatible with this MP. You can only use this option once in the command.

- **agent_version**

  Minimum Enterprise Manager Agent version compatible with this MP. You can only use this option once in the command.

- **file**

  Type and path of other files to be included in the MP. You can specify this option more than once. The supported types are:

  - `MONITORING_BINARY` — Monitoring binary or executable the target-type definition uses to collect data.
  - `MONITORING_SCRIPT` — Monitoring script the target-type definition uses to collect data.
  - `REPORT_DEFINITION` — PL/SQL calls into the reporting framework to define reports for this version of the MP.
  - JOB_SCRIPT — To be provided.
  - `JOB_DEFINITION` — To be provided.

  You must specify `POLICY_DEPLOY` and `POLICY_UNDEPLOY` together.

- **func_desc**

  Describes the purpose of the MP and any other general information about the MP. You can only use this option once in the command.

- **req_desc**

  Describes any conditions that may exist for this MP to be successfully deployed and used. This description is optional, so you can ignore it, but this is not recommended. You can only use this option once in the command.

Example

The following example adds Management Plug-in files to a Management Plug-in Archive called `my_new_type.jar`.

```
emcli add_mp_to_mpa
      -mpa="/my_dir/my_new_type.jar"
      -mp_version="2.0"
      -ttd="/my_dir/ttd/new_type.xml"
      -dc="/my_dir/dc/new_type.xml"
      -file="MONITORING_SCRIPT:/my_dir/script1.pl"
      -file="MONITORING_SCRIPT:/my_dir/script2.pl"
      -file="MONITORING_BINARY:/my_dir/bin1"
      -func_desc="Management Plug-in to define target type new_type"
```

add_target

Adds a target to be monitored by Enterprise Manager. The target type specified is checked on the Management Agent for existence and for required properties, such as user name and password for host target types, or log-in credentials for database target types. You must specify any required properties of a target type when adding a new target of that type.

For `oracle_database` target types, you must specify Role with the monitoring credentials. If the Role is Normal, the UserName must be `dbsnmp`. Otherwise, the Role must be SYSDBA, and UserName can be any user with SYSDBA privileges.

Format

```
add_target
      -name="name"
      -type="type"
      -host="hostname"
      [-properties="pname1:pval1;pname2:pval2;..."]
      [-credentials="userpropname:username;pwdpropname:password;..."]
      [-input_file="parameter_tag:file_path"]
      [-display_name="display name"]
      [-groups="groupname1:grouptype1;groupname2:grouptype2;..."]
      [-timezone_region="gmt offset"]
      [-monitor_mode="monitor mode"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name. Names cannot contain colons (:), semi-colons (;), or any leading or trailing blanks.

- **type**

  Target type. Standard target types include: `host`, `oracle_database`, `oracle_apache`, `oracle_listener`, and `oracle_emd`. To see all available target types available for your environment, check the `$AGENT_HOME/sysman/admin/metadata` directory. A metadata file (XML) exists for each target type.

- **host**

  Network name of the machine running the Management Agent that is collecting data for this target instance.

- **properties**

  Name-value pair (that is, `prop_name:prop_value`) list of properties for the target instance. The "name"(s) are identified in the target-type metadata definition. They must appear exactly as they are defined in that file. Metadata files are located in `$AGENT_HOME/sysman/admin/metadata`.

- **credentials**

  Monitoring credentials (name-value pairs) for the target instance. The "name"(s) are identified in the target-type metadata definition as credential properties. The credentials must be specified exactly as they are defined in the target's metadata file. Metadata files are located in `$AGENT_HOME/sysman/admin/metadata`.

- **input_file**

  Used in conjunction with the `-credentials` option, this option allows you to store specific target monitoring credential values, such as passwords, in a separate file. The `-input_file` option specifies a mapping between a tag and a local file path. The tag is specified in lieu of specific monitoring credentials of the `-credentials` option. The tag must not contain colons (:) or semi-colons (;).

- **display_name**

  Target name that is displayed in the Enterprise Manager Grid Control console.

- **groups**

  Name-value pair list of the groups to which this target instance belongs. Follows the format of `groupname:grouptype;groupname2:grouptype2`.

- **timezone_region**

  GMT offset for this target instance (-7 or -04:00 are acceptable formats).

- **monitor_mode**

  Either 0, 1, or 2 (default is 0). 1 indicates OMS-mediated monitoring and 2 indicates agent-mediated monitoring.

Examples

The following example adds an `oracle_database` target with the name "database." Note how the credentials are specified. The "name"(s) in the name-value pairs come from the `oracle_database` metadata file. They must appear exactly as they are named in that file. This also applies for the property "name"(s). This example uses the base minimum of required credentials and properties for the database target.

```
emcli add_target
      -name="database"
      -type="oracle_database"
      -host="myhost.us.oracle.com"
      -credentials="UserName:dbsnmp;password:dbsnmp;Role:Normal"
      -properties="SID:semcli;Port:15091;OracleHome:/oracle;
MachineName:smpamp-sun1.us.oracle.com"
      -groups="Group1:database_group;Group2:group"
```

The following example adds an `oracle_database` target with the name "database." This example illustrates the use of the `input_file` to camouflage the credentials. The password is actually in a file named `at_pwd_file`. The `input_file` argument is used to replace `PWD_FILE` with the contents of the `at_pwd_file` in the credentials argument.

```
emcli add_target
      -name="database"
      -type="oracle_database"
      -host="myhost.us.oracle.com"
      -credentials="UserName:dbsnmp;password:PWD_FILE;Role:Normal"
      -properties="SID:semcli;Port:15091;OracleHome:/oracle;
MachineName:smpamp-sun1.us.oracle.com"
      -input_file="PWD_FILE:/emcli_dir/pwdfiles/at_pwd_file"
```

The following example adds an `oracle_listener` target with the name "mylist". The `LsnrName` is the name of the listener as configured in the `listener.ora` file and `ListenerOraDir` is the directory containing the `listener.ora` file.

```
emcli add_target
```

```
      -name="mylist"
      -type="oracle_listener"
      -host="myhost.us.oracle.com"
      -properties="LsnrName:LISTENER;ListenerOraDir:/oracle/lsnr;
Port:15091;OracleHome:/oracle;Machine:smpamp-sun1.us
```

apply_privilege_delegation_setting

Activates Sudo or PowerBroker settings for specified targets.

Format

```
emcli apply_privilege_delegation_setting
        -setting_name="setting_name"
        [-target_names="name1;name2;..."]
        -target_type="host"
        [-input_file="FILE:file_path"]
        [-force="yes/no"]

[ ] denotes that the parameter is optional
```

Options

- **setting_name**

  Name of the setting you want to apply.

- **target_names**

  List of target names. The newly submitted setting applies to this list of EM targets.

  – All targets must be of the same type.

  – The target list must not contain more than one element if the element's target type is "group."

  – The group referenced above should have at least one host target.

- target_type

  Type of targets to which the setting is applied. Valid target types are "host" or "composite" (group).

- **input_file**

  Path of the file that has target names. This option enables you to pass targets in a separate file. The file cannot contain any colons (`:`) or semi-colons (`;`).

- force

  If `yes`, the operation continues and ignores any invalid targets. The default is `no`.

Examples

The following example applies a privilege setting named `sudo_setting`. This setting applies to targets of type `host`, and it is being applied to `host1`, `host2`, and so forth.

```
emcli apply_privilege_delegation_setting
      -setting_name=sudo_setting
      -target_type=host
      -target_names="host1;host2;....."
```

The following example applies a privilege setting named `sudo_setting`. This setting applies to targets of type `host`, and it is being applied to `host1`, `host2`, and so forth. The `force` flag indicates that the setting is applied to all valid targets, and invalid targets are ignored.

```
emcli apply_privilege_delegation_setting
      -setting_name=sudo_setting
      -target_type=host
```

```
      -target_names="host1;host2;....."
      -force=yes
```

The following example applies a privilege setting named `sudo_setting`. This setting applies to targets of type `host`, and host names are selected from `/home/jdoe/file.txt` (one host per line). The `force` flag indicates that the setting is applied to all valid targets, and invalid targets are ignored.

```
emcli apply_privilege_delegation_setting
      -setting_name=sudo_setting
      -target_type=host
      -input_file="FILE:/home/jdoe/file.txt"
      -force=yes
```

apply_template

Applies a template to a list of specified targets. The parameters to the verb can be supplied in any order.

Format

```
emcli apply_template
        -name="template_name"
        -targets="tname1: ttype1;tname2: ttype2;..."
        [-copy_flags="0" or "1" or "2"]
        [-input_file="FILE1:file_name"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Template name as it exists in the database. Names cannot contain colons (:), semi-colons (;), or any leading or trailing blanks.

- **targets**

  The targets should be specified in the following sequence:

  TargetName1:TargetType1;TargetName2:TargetType2

  For example:

  `db1:oracle_database;my db group:composite`

  (semi colon) is the target separator. Ideally, non-composite targets should be of the target type applicable to the template. If not, the template is not applied to the said target. For composite targets, the template is applied only to the member targets that belong to the target type for which the template is applicable.

- **copy_flags**

  This applies only for the metrics with multiple thresholds.

  '0' indicates: Apply threshold settings for key values common to the template and target.

  '1' indicates: Remove key value threshold settings in the target and replace them with key value threshold settings from the template.

  '2' indicates: Apply threshold settings for all key values defined in the template. The default is option '0'.

- **input_file**

  The file should contain the credentials according to the following cases:

  One:

  If User-Defined Metric (UDM) credentials take the value "All User-Defined Metrics use the same credentials," you need to provide these credentials in the following format:

  Sample input file:

  ```
  credListType:all;
  usr_name:joe1;passwrd:pass1;
  ```

Two:

If UDM Credentials take the value "Each User-Defined Metric uses its own credentials,"you need to provide these credentials in the following format:

Sample input file:

```
credListType:perUDM;
udm_name:UDM1;usr_name:joe1;passwrd:pass1;
udm_name:UDM2;usr_name:joe2;passwrd:pass2;
```

Three:

If UDM Credentials take the value "Each User-Defined Metric uses different credentials for different targets," you need to provide these credentials in the following format:

Sample input file:

```
credListType:perTargetperUDM;
udm_name:UDM1;tgt_name:TNAME1;usr_name:joe1;passwrd:pass1;
udm_name:UDM1;tgt_name:TNAME2;usr_name:joe2;passwrd:pass2;
udm_name:UDM2;tgt_name:TNAME1;usr_name:joe3;passwrd:pass3;
udm_name:UDM2;tgt_name:TNAME2;usr_name:joe4;passwrd:pass4;
```

It is important that you specify the "credListType" in every input text file that you specify.

Examples

The following example applies a monitoring template named `my_db_template`. This template applies to targets of type `oracle_database`, and it is being applied to `db1` which is of type `oracle_database` and `my_db_group` which is of type `composite`. For composite targets, the template is only applied to member targets that belong to the target type for which the template is applicable. Since the `copy_flags` option is not specified, the default option ("Apply threshold settings for monitored objects common to both template and target") is meant.

```
emcli apply_template -name="my_db_template"
          -targets="db1:oracle_database;my_db_group:composite"
```

The following example applies a monitoring template named `my_db_template`. This template applies to targets of type `oracle_database` and it is being applied to `db1` which is of type `oracle_database` and `my_db_group` which is of type `composite`. In case of composite targets, the template is applied only to member targets that belong to the target type for which the template is applicable. In this case, since the `copy_flags` option is specified as 1, the threshold settings on the target will be duplicated.

```
emcli apply_template -name="my_db_template"
          -targets="db1:oracle_database;my_db_group:composite"
          -copy_flags="1"
```

The following example applies a monitoring template named `my_db_template`. This template applies to targets of type `oracle_database` and it is being applied to `db1` which is of type `oracle_database` and `my_db_group` which is of type `composite`. For composite targets, the template is applied only to member targets that belong to the target type for which the template is applicable. In this case, since the `copy_flags` option is specified as "1", the threshold settings on target will be duplicated.

Furthermore, the credentials needed for the UDMs are present in the file "/usr/vmotamar/db_credentials.txt".

```
emcli apply_template -name="my_db_template"
          -targets="db1:oracle_database;my_db_group:composite"
          -copy_flags="1" -input_file= "FILE1:/usr/vmotamar/db_credentials.txt"
```

apply_template_tests

Applies the variables and test definitions from the file(s) into a repository target.

Format

```
apply_template_tests
      -targetName=<target name>
      -targetType=<target type>
      -input_file=template:<template filename>
      [-input_file=variables:<variable filename>]
      [-overwriteExisting=<all | none | <test1>:<type1>;<test2>:<type2>;...>]
      [-encryption_key=<key>]

[ ] denotes that the parameter is optional
```

Options

- **targetName**

  Target name.

- **targetType**

  Target type.

- **input_file**

  Name of the input file containing the test definitions.

- **input_file**

  Name of the input file containing the variable definitions. If this attribute is not specified, the variables are pulled from the same file containing the test definitions.

  The variables file format is as follows:

  ```
  <variables xmlns="template">
  <variable name="<name1>" value="<value1>"/>
  <variable name="<name2>" value="<value2>"/>
  ...
  </variables>
  ```

- **overwriteExisting**

  Specifies which tests should be overwritten in case they already exist on the target. The possible values are:

  1. 'none' (default): None of the existing tests on the target will be overwritten.
  2. 'all': If a test with the same name exists on the target, it will be overwritten with the test definition specified in the template file.
  3. <test1>:<type1>;<test2>:<type2>;...: If any of tests with names <test1>, <test2>, etc. exist on the target, they will be overwritten with the definition in the template file.

- **encryption_key**

  Optional key to decrypt the file contents. This key should be the same as the one used to encrypt the file.

Examples

The following example applies the test definitions contained in file my_template.xml into the Generic Service target my_target, using key my_password to decrypt the file contents. If tests with names my_website or my_script exist on the target, they will be overwritten by the test definitions in the file.

```
emcli apply_template_tests
      -targetName='my_target' -targetType='generic_service'
      -input_file=template:'my_template.xml' -encryption_key='my_password'
      -overwriteExisting='my_website:HTTP;my_script:OS'
```

argfile

Executes a single EM CLI Verb where both Verb and arguments are contained in an ASCII file. This Verb allows you to use verbs with greater flexibility. For example, when specifying a large list of targets to be blacked out (`create_blackout` Verb), you can use the `argfile` Verb to input the target list from a file.

> **Note:** The ASCII file can only operate using a single Verb. Using more than one Verb in the file causes execution errors.

Format

```
argfile /path/to/<file_name>
```

Options

None.

Examples

```
emcli argfile my_verb_arguments
```

assign_test_to_target

Assigns a test-type to a target-type. If a test-type t is assigned to target-type T, all targets of type T can be queried with tests of type t.

Format

```
assign_test_to_target
      -testtype=test-type to be assigned
      -type=target type
      [-tgtVersion]=version of target type

[ ] denotes that the parameter is optional
```

Options

- **testtype**

  Test-type to be assigned. Should be the internal name; that is, 'HTTP' instead of 'Web Transaction'.

- **type**

  Service target type.

- **tgtVersion**

  Version of the target type. If not specified, the latest version is used.

Examples

The following example assigns test type HTTP to targets of type generic service v2.

```
emcli assign_test_to_target -testtype='HTTP' -type='generic_service'
      -tgtVersion='2.0'
```

change_service_system_assoc

Changes the system that hosts a given service.

Format

```
change_service_system_assoc
      -name='name'
      -type='type'
      -systemname='system name'
      -systemtype='system type'
      -keycomponents='keycomp1name:keycomp1type[;keycomp2name:keycomp2type;...]'

[ ] denotes that the parameter is optional
```

Options

- **name**

  Service name.

- **type**

  Service type.

- **systemname**

  System on which service resides.

- **systemtype**

  System type.

- **keycomponents**

  Name-type pair (such as `keycomp_name:keycomp_type`) list of key components in the system that are used for the service.

Examples

The following example changes system for a generic service named `my service` to a generic system named `my system` with specified key components.

```
emcli change_service_system_assoc
      -name='my service' -type='generic_service'
      -systemname='my system' -systemtype='generic_system'
      -keycomponents='database:oracle_database; mytestbeacon:oracle_beacon'
```

clone_as_home

Clones the specified Application Server Oracle Home or S/W Library component from the target host to specified destinations. For a Portal and Wireless installation, the OID user and password are also needed. For a J2EE instance connected to only a DB-based repository, a DCM Schema password is needed.

Passing Variables Through EMCLI

When working with variables such as `%perlbin%` or `%oracle_home%`, EM CLI passes variable values from the current local environment instead of the variables themselves. To pass variables through an EM CLI command, as might be the case when using the `-prescripts` or `-postscripts` options, you can place the EM CLI command in a batch file and replace all occurrences of `%` with `%%`.

Format

```
emcli clone_as_home
       -input_file="dest_properties:file_path"
       -list_exclude_files="list of files to exclude"
       -isSwLib="true/false"
       -tryftp_copy="true/false"
       -jobname="name of cloning job"
       -iasInstance=instance
       -oldIASAdminPassword=oldpass
       -newIASAdminPassword=newpass
       [-oiduser=oid admin user]
       [-oidpassword=oid admin password]
       [-dcmpassword=dcm schema password]
       [-prescripts="script name to execute"]
       [-run_prescripts_as_root="true/false"]
       [-postscripts="script to execute"]
       [-run_postscripts_as_root="true/false"]
       [-rootscripts="script name to execute"]
       [-swlib_component ="path:path to component;version:rev"]
       [-source_params="TargetName:name;HomeLoc:loc;HomeName:name;
         ScratchLoc:Scratch dir Location"
       [-jobdesc="description"]

[ ] denotes that the parameter is optional
```

Options

- **dest_properties**

  File containing information regarding the targets.

  Each line in the file corresponds to information regarding one destination.

  Format:

  `Destination Host Name1;Destination Home Loc; Home Name; Scratch Location;`

- **list_exclude_files**

  Comma-separated list of files to exclude. Not required if the source is software lib. "*" can be used as a wild card.

- **isSwLib**

  Specifies whether it is an Oracle Home database or Software Library.

- **ryftp_copy**

  Try FTP to copy or not. You should set the FTP copy option to false when using EM CLI from the command line.

- **jobname**

  Name of the cloning job.

- **iasInstance**

  Name of instance.

- **oldIASAdminPassword**

  Old Application Server administrator password.

- **newIASAdminPassword**

  New Application Server administrator password.

- **oiduser**

  OID admin user.

- **oidpassword**

  OID admin password.

- **dcmpassword**

  DCM schema password.

- **prescripts**

  Path of script to execute.

  > **Note:** Double-quoted parameters can be passed using an escape (\) sequence. For example:
  >
  > ```
  > prescripts=" <some value here>=\"some value here\" "
  > ```

- **run_prescripts_as_root**

  Run prescripts as "root". By default, the option is set to false.

- **postscripts**

  Path of script to execute.

- **run_postscripts_as_root**

  Run postscripts as "root". By default, the option is set to false.

- **rootscripts**

  Path of the script to execute. The job system environment variables (`%oracle_home%`, `%perl_bin%`) can be used for specifying script locations.

- **swlib_component**

  Path to the Software Library to be cloned. "isSwLib" must be true in this case.

- **source_params**

  Source Oracle home information. "isSwLib" must be false in this case.

- **jobdesc**

  Description of the job. If not specified, a default description is generated automatically.

clone_crs_home

Creates an Oracle Clusterware cluster given a source Clusterware home location or a Clusterware S/W Library component for specified destination nodes.

Format

```
emcli clone_crs_home
      -input_file="dest_properties:file_path"
      -list_exclude_files="list of files to exclude"
      -isSwLib="true/false"
      -tryftp_copy="true/false"
      -jobname="name of cloning job"
      -home_name="name of home to use when creating Oracle Clusterware cluster"
      -home_location="location of home when creating Oracle Clusterware cluster"
      -clustername=name of cluster to create
      -ocrLoc=ocr location
      -vdiskLoc=voting disk location
      [-prescripts="script name to execute"]
      [-run_prescripts_as_root="true/false"]
      [-postscripts="script to execute"]
      [-run_postscripts_as_root="true/false"]
      [-rootscripts="script name to execute"]
      [-swlib_component ="path:path to component;version:rev"]
      [-source_params="TargetName:name;HomeLoc:loc;HomeName:name;
        ScratchLoc:Scratch dir Location"]
      [-jobdesc="description"]

[ ] denotes that the parameter is optional
```

Options

- **dest_properties**

  File containing information regarding the targets.

  Each line in the file corresponds to information regarding one destination.

  Format:

  `Destination Host Name;Destination Node Name;Scratch Location;PVTIC;VirtualIP;`

- **list_exclude_files**

  Comma-separated list of files to exclude. Not required if the source is software lib. An asterisk "*" can be used as a wildcard.

- **isSwLib**

  Specifies whether it is an Oracle Home database or Software Library.

- **tryftp_copy**

  Try ftp to copy or not. You should set the ftp copy option to false when using `emcli` from the command line.

- **jobname**

  Name of the cloning job.

- **home_name**

  Name of the home to use for all homes in the Oracle Clusterware cluster.

- **home_location**

  Location of the home to use for all homes in the Oracle Clusterware cluster.
- **clustername**

  Name of the cluster to create.
- **ocrLoc**

  Oracle Cluster Registry Loaction.
- **vdiskLoc**

  Voting disk location.
- **prescripts**

  Path of the script to execute.

  > **Note:** Double-quoted parameters can be passed using an escape (\) sequence. For example:
  >
  > ```
  > prescripts=" <some value here>=\"some value here\" "
  > ```

- **run_prescripts_as_root**

  Run prescripts as "root". By default, this option is set to false.
- **postscripts**

  Path of the script to execute.
- **run_postscripts_as_root**

  Run postscripts as "root". By default, it is false.
- **rootscripts**

  Path of the script to execute.
- **swlib_component**

  Path to the Software Library to be cloned. "isSwLib" must be true in this case.
- **source_params**

  Source Oracle home info. "isSwLib" must be false in this case.
- **jobdesc**

  Description of the job. If not specified, a default description is generated automatically.

Examples

```
emcli clone_crs_home -input_file="dest_properties:crs.prop"  -isSwLib="true"
  -tryftp_copy="true" -jobname="crs cloning job2" -home_name="cloneCRS1"
  -home_location="/scratch/scott/cloneCRS1 " -clustername="crscluster"
  -ocrLoc="/scratch/shared/ocr" -vdiskLoc="/scratch/shared/vdisk"
  -postscripts="%perlbin%/perl%emd_root%/admin/scripts/cloning/samples/post_crs
_  create.pl ORACLE_HOME=%oracle_home%"
  -run_postscripts_as_root="true" -rootscripts="%oracle_home%/root.sh"
  -swlib_component="path:Components/crscomp;version:.1"
```

Passing Variables Through EMCLI

When working with variables such as `%perlbin%` or `%oracle_home%`, EM CLI passes variable values from the current local environment instead of the variables themselves. To pass variables through an EM CLI command, as might be the case when using the `-prescripts` or `-postscripts` options, you can place the EM CLI command in a batch file and replace all occurrences of `%` with `%%`.

clone_database_home

Clones the specified Oracle Home or S/W Library from the target host to specified destinations. If the isRac option is true, a RAC cluster is created. If the isRac option is true, the home name and location of the RAC cluster are needed

Format

```
emcli clone_database_home
      -input_file="dest_properties:file_path"
      -list_exclude_files="list of files to exclude"
      -isSwLib="true/false"
      -isRac="true/false"
      -tryftp_copy="true/false"
      -jobname="name of cloning job"
      [-home_name="name of home to use when creating RAC cluster"]
      [-home_location="location of home when creating RAC cluster"]
      [-prescripts="script name to execute"]
      [-run_prescripts_as_root="true/false"]
      [-postscripts="script to execute"]
      [-run_postscripts_as_root="true/false"]
      [-rootscripts="script name to execute"]
      [-swlib_component ="path:path to component;version:rev"]
      [-source_params="TargetName:name;HomeLoc:loc;HomeName:name;
        ScratchLoc:Scratch dir Location"
      [-jobdesc="description"]

[ ] denotes that the parameter is optional
```

Options

- **dest_properties**

  File containing information regarding the targets.

  Each line in the file corresponds to information regarding one destination.

  Format if cloning a database (isRac is false):

  `Destination Host Name1;Destination Home Loc; Home Name; Scratch Location;`

  Format if cloning a RAC cluster (isRac is true):

  `Host Name;Node Name;Scratch Location;`

- **list_exclude_files**

  Comma-separated list of files to exclude. This is not required if the source is software lib. "*" can be used as a wild card.

- **isSwLib**

  Specifies whether the source is an Oracle Home database or Software Library.

- **isRac**

  Specifies whether cloning in RAC mode.

- **tryftp_copy**

  Try FTP to copy or not. You should set the FTP copy option to false when using EM CLI from the command line.

- **jobname**

  Name of the cloning job.
- **home_name**

  Name of the home to use when creating a RAC cluster.
- **home_location**

  Location of the home to use when creating a RAC cluster.
- **prescripts**

  Path of the script to execute.

  Note: Double-quoted parameters can be passed using an escape (\) sequence. For example:

  ```
  prescripts=" <some value here>=\"some value here\" "
  ```

- **run_prescripts_as_root**

  Run prescripts as "root". By default, it is false.
- **postscripts**

  Path of the script to execute.
- **run_postscripts_as_root**

  Run postscripts as "root". By default it is false.
- **rootscripts**

  Path of the script to execute. The job system environment variables (`%oracle_home%`, `%perl_bin%)` can be used for specifying script locations.
- **swlib_component**

  Path to the Software Library to be cloned. "isSwLib" must be true in this case.
- **source_params**

  Source Oracle home info. "isSwLib" must be false in this case.
- **jobdesc**

  Description of the job. If not specified, it is automatically generated.

Examples

```
emcli clone_database_home
      -input_file="dest_properties:clonedestinations"
      -list_exclude_files="*.log,*.dbf,sqlnet.ora,tnsnames.ora,listener.ora"
      -isSwLib="false"
      -isRac="false"
      -tryftp_copy="false"
      -jobname="clone database home"
      -prescripts="/home/joe/myScript"
      -run_prescripts_as_root="true"
      -rootscripts="%oracle_home%/root.sh"
      -source_params="TargetName:host.domain.com;HomeLoc=/oracle/database1;
HomeName=OUIHome1;ScratchLoc=/tmp"
```

Passing Variables Through EMCLI

When working with variables such as `%perlbin%` or `%oracle_home%`, EM CLI passes variable values from the current local environment instead of the variables themselves. To pass variables through an EM CLI command, as might be the case when using the `-prescripts` or `-postscripts` options, you can place the EM CLI command in a batch file and replace all occurrences of `%` with `%%`.

create_aggregate_service

Defines an aggregate service: name and its sub-services. After the aggregate service is created, you can edit it from the Enterprise Manager Grid Control console to configure performance and usage metrics to be collected and displayed.

Format

```
create_aggregate_service
      -name="name"
      -type="type"
      -add_sub_services="name1:type1;name2:type2;..."
      -avail_eval_func="function to evaluate availability"
      [-timezone_region="timezone region"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Aggregate service name.

- **type**

  Aggregate service type.

- **add_sub_services**

  Sub-services list.

- **avail_eval_func**

  PL/SQL function to evaluate the availability of the aggregate service. Use [or|and] for a predefined evaluate helper function.

- **timezone_region**

  Time Zone Region of the service.

Examples

```
emcli create_aggregate_service -name="My_Name"
      -type="aggregate_service"
      -add_sub_services="sub1:type1;sub2:type2"
      -avail_eval_func="my_pkg.my_eval_func"
      -timezone_region="PST"
```

create_blackout

Creates a scheduled blackout to suspend any data collection activity on one or more monitored targets.

Format

```
create_blackout
      -name="name"
      add_targets="name1:type1;name2:type2;..."...
      reason="reason"
      -description="description"]
      -jobs_allowed]
      -propagate_targets]
      schedule=
         frequency:<once|interval|weekly|monthly|yearly>];
         duration:[HH...][:mm...];
         [start_time:<yy-MM-dd HH:mm>];
         [end_time:<yy-MM-dd HH:mm>];
         [repeat:<#m|#h|#d|#w>];
         [months:<#,#,...>];
         [days:<#,#,...>];
         [tzinfo:<specified|target|repository>]
         [tzoffset:#|[-][HH][:mm]]

[ ] denotes that the parameter is optional
```

Constraints on schedule arguments:

```
frequency:once
    requires => duration or end_time
    optional => start_time, tzinfo, tzoffset
frequency:interval
    requires => duration, repeat
    optional => start_time, end_time, tzinfo, tzoffset
frequency:weekly
    requires => duration, days
    optional => start_time, end_time, tzinfo, tzoffset
frequency:monthly
    requires => duration, days
    optional => start_time, end_time, tzinfo, tzoffset
frequency:yearly
    requires => duration, days, months
    optional => start_time, end_time, tzinfo, tzoffset
```

Options

- **name**

  Name of the blackout to create.

- **add_targets**

  Targets to add to the blackout, each specified as `target_name:target_type`. The `-add_targets` option can be specified more than once.

- **reason**

  Reason for the blackout. If you have `SUPER_USER` privileges (you are an Enterprise Manager Super Administrator), any text string can be used for the reason. The reason is added to the list of allowable blackout reasons if it is not already in the list. If you do not have `SUPER_USER` privileges, you must specify one of the text strings returned by the `get_blackout_reasons` Verb.

- **description**

  Description or comments pertaining to the blackout. The description, limited to 2000 characters, can be any text string.

- **jobs_allowed**

  When this option is specified, jobs are allowed to run against blacked out targets during the blackout period. When this option is not specified, jobs scheduled to be run against these targets are not allowed to run during the blackout period. After a blackout has been created, you cannot change the "allowed jobs" option from either EM CLI or the Enterprise Manager Grid Control console.

- **propagate_targets**

  When this option is specified, a blackout for a target of type "host" applies the blackout to all non-agent targets on that host. Regardless of whether this option is specified, a blackout for a target that is a composite or a group applies the blackout to all members of the composite or group.

- **schedule**

  Blackout schedule. Note that the "frequency" argument determines which other arguments are required or optional.

- **schedule=frequency**

  Type of blackout schedule (default is "once").

- **schedule=duration**

  Duration in hours and minutes of the blackout (-1 means indefinite). Hours and minutes each can be up to 6 digits long.

- **schedule=start_time**

  Start date/time of the blackout. The default value is the current date/time. The format of the value is "yy-MM-dd HH:mm", for example: "2003-09-25 18:34"

- **schedule=end_time**

  Last date/time of the blackout. When "frequency" is weekly, monthly, or yearly, only the date portion is used. When "frequency" is interval or once, the date and time are taken into account. The format of the value is "yy-MM-dd HH:mm"; for example: "2003-09-25 18:34"

- **schedule=repeat**

  Time between successive start times of the blackout. The letter following the number value represents the time units: "m" is minutes, "h" is hours, "d" is days, and "w" is weeks.

- **schedule=months**

  List of integer month values in the range 1-12. Each value must have a corresponding "day" value to fully specify (month, day) pairs that indicate the blackout starting days of the year.

- **schedule=days**

  When "frequency" is weekly, this is a list of integer day-of-week values in the range 1-7 (1 is Sunday). When "frequency" is monthly, this is a list of integer day-of-month values in the range 1-31 or -1 (last day of the month). When "frequency" is yearly, this is a list of integer day-of-month values in the range 1-31 or -1 (last day of the month); in this case, the month is taken as the corresponding "month" value for each (month, day) pair.

- **schedule=tzinfo**

  Type of timezone. The `tzinfo` argument is used in conjunction with `tzoffset`. Available timezone types are: "specified" (offset between GMT and the target timezone), "target" (timezone of the specified target), and "repository" (repository timezone -- default setting when tzinfo is not specified). See `-schedule=tzoffset` for more information.

- **schedule=tzoffset**

  Value of the timezone. When the `tzinfo` argument is not specified or is "repository", the timezone value is the repository timezone. In this case, the `tzoffset` argument must not be specified. Otherwise, the `tzoffset` argument is required. When `tzinfo` is set to "specified", the `tzoffset` argument specifies the offset in hours and minutes between GMT and the timezone. When `tzinfo` is set to "target", the `tzoffset` argument specifies an integer index (the first is 1) into the list of targets passed as arguments. For example, for a `tzoffset` setting of 1, the timezone of the first target specified in the `-add_targets` option is used.

  Note that the timezone is applied to the start time and the end time of the blackout periods. The timezones associated with each target are not taken into account when scheduling the blackout periods (except that when `tzinfo` is set to "target", the specified target's timezone is used for the blackout times).

Examples

The following example creates blackout `b1` for the specified target (`database2`) to start immediately and last for 30 minutes.

```
emcli create_blackout -name=b1 -add_targets=database2:oracle_database
      -schedule="duration::30"
      -reason="good reason1"
```

The following example creates blackout `b1` for all targets on `myhost` to start immediately and last until 2007-04-26 05:00 (in the timezone GMT-4hours).

```
emcli create_blackout -name=b1 -add_targets=myhost:host
      -propagate_targets -jobs_allowed
      -schedule="end_time:2007-04-26 05:00;tzinfo:specified;tzoffset:-4"
      -reason="good reason2"
```

The following example creates blackout `b1` for all targets in group `mygroup` to start immediately and last until 2007-04-26 05:00 (in the timezone GMT-4hours). No jobs are allowed to run during the blackout.

```
emcli create_blackout -name=b1 -add_targets=mygroup:group
      -schedule="end_time:2007-04-26 05:00;tzinfo:specified;tzoffset:-4"
      -reason="good reason3"
```

The following example creates blackout `b1` for the specified targets (`database2` and `database3`) to start at 2007-08-24 22:30 and last for 30 minutes. The timezone is the timezone for the `database2` target.

```
emcli create_blackout -name=b1
      -add_targets="database2:oracle_database;database3:oracle_database
      -schedule="frequency:once;start_time:07-08-24
22:30;duration::30;tzinfo:target:tzoffset:1"
      -reason="good reason4"
```

The following example creates blackout `b1` for the specified targets (`database2` and `database3`) to start at 2007-08-24 22:30 and last for 30 minutes. The timezone is the timezone for the `database3` target.

```
emcli create_blackout -name=b1 -add_targets=database2:oracle_database
      -add_targets=database3:oracle_database
      -schedule="frequency:once;start_time:07-08-24
22:30;duration::30;tzinfo:target;tzoffset:2"
      -reason="good reason5"
```

The following example creates blackout `b2` for the specified target (`database2`) to start at 2007-08-25 03:00 and every day thereafter, and to last 2 hours each time. The timezone is the repository timezone.

```
emcli create_blackout -name=b2 -add_targets=database2:oracle_database
      -schedule="frequency:interval;start_time:2007-08-25
03:00;duration:2;repeat=1d"
      -reason="good reason"
```

The following example creates blackout `b2` for the specified target (`database2`) to start immediately and every 2 days thereafter (until 06-12-31 23:59), and to last 2 hours 5 minutes each time. The timezone is the repository timezone.

```
emcli create_blackout -name=b2 -add_targets=database2:oracle_database
      -schedule="frequency:interval;duration:2:5;end_time:06-12-31
23:59;repeat=2d;tzinfo:repository"
      -reason="another good reason"
```

The following example creates blackout `b4` for all targets on `myhost` and `otherhost` to start every Sunday through Thursday at the current time. The blackout will last 1 hour each time.

```
emcli create_blackout -name=b4 -add_targets="myhost:host;otherhost:host"
      -propagate_targets
      -schedule="frequency:weekly;duration:1:;days=1,2,3,4,5"
      -reason="very good reason"
```

The following example creates blackout `b5` for all targets within group `mygroup` to start on the 15th and last day of each month at time 22:30 and last until 2006-12-24 (2006-12-15 will be the actual last blackout date). The blackout will last 1 hour 10 minutes each time. Jobs are allowed to run during the blackouts.

```
emcli create_blackout -name=b5 -add_targets=mygroup:group
      -propagate_targets -jobs_allowed
      -schedule="frequency:monthly;duration:1:10;start_time:06-10-24 22:30;end_
time:06-12-24 23:59:days=15,-1"
      -reason="pretty good reason"
```

The following example creates blackout `b6` for the specified target (`database2`) to start at 13:30 on the following dates of each year: 03-02, 04-22, 09-23. The blackout will last 2 hours each time. Jobs are not allowed to run during the blackouts.

```
emcli create_blackout -name=b6 -add_targets=database2:oracle_database
      -propagate_targets
      -schedule="frequency:yearly;duration:2;start_time:07-08-24
13:30:months=3,4,9;days=2,22,23"
      -reason="most excellent reason"
```

create_group

Defines a group: name and its members. After the group is created, you can edit the group from the Enterprise Manager Grid Control console to configure Summary Metrics to be displayed for group members.

Format

```
create_group
      -name="name"
      [-type=<group>]
      [-add_targets="name1:type1;name2:type2;..."]...

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the group.

- **type**

  Group type: group. Defaults to "group".

- **add_targets**

  Add existing targets to the group. Each target is specified as a name-value pair `target_name:target_type`. The `-add_targets` option can be specified more than once.

Examples

The following example creates a database-only group named `db_group`. This group consists of two Oracle databases: `emp_rec` and `payroll`.

```
emcli create_group -name=db_group
      -add_targets="emp_rec:oracle_database"
      -add_targets="payroll:oracle_database"
```

The following example creates a mixed member type group named `my_group` that consists of an oracle database (`database2)`, listener (`dblistener`), and host (`mymachine.myco.com`).

```
emcli create_group -name=my_group
      -add_targets="database2:oracle_database;dblistener:oracle_listener
      -add_targets="mymachine.myco.com:host"
```

The following example creates a host-only group named `my_hosts` that consists of three machines within the `oracle.com` domain: `smpsun`, `dlsun`, and `supersun`.

```
emcli create_group -name=my_hosts
      -add_targets="smpsun.oracle.com:host"
      -add_targets="dlsun.oracle.com:host;supersun.oracle.com:host"
```

create_privilege_delegation_setting

Creates Sudo or PowerBroker settings to apply later. You must create at least one setting to use the `apply_privilege_delegation_setting` verb.

Format

```
emcli create_privilege_delegation_setting
        -setting_name="setting_name"
        -setting_type="SUDO/POWERBROKER"
        [-settings="Sudo/Powerbroker setting value"]
        [-separator=settings=";"]
        [-subseparator=settings=","]
        [-disabled="yes/no"]

[ ] denotes that the parameter is optional
```

Options

- **setting_name**

  Name of the setting.

- **setting_type**

  Type of setting you want to create.

- settings

  Parameter value. Do not include this if the `disabled` flag is set to `yes`.

- **separator**

  Delimiter inserted between name-value pairs for the given option name. The default value is a semi-colon (`;`).

- **subseparator**

  Separator inserted between the name and value in each name-value pair for the given option name. The default value is a semi-colon (`;`).

- disabled

  Status of the setting. You can create a disabled setting to remove settings from targets. The default value is `no`.

Examples

The following example creates a setting named `sudo_setting`. The setting is of type SUDO, and the Sudo path used is `/usr/local/bin/sudo`. Sudo arguments are:

-S
-u
%RUNAS%
%command%

```
emcli create_privilege_delegation_setting
      -setting_name=sudo_setting
      -setting_type=SUDO
      -settings="SETTINGS:/usr/local/bin/sudo -S -u %RUNAS% %command%"
```

The following example creates a setting named `pb_setting`. The setting is of type POWERBROKER, and the PowerBroker path used is `/etc/pbrun`. Arguments are:

```
%RUNAS%
%PROFILE%
%command%
;PASSWORD_ PROMPT_STRING
Password:
```

```
emcli create_privilege_delegation_setting
      -setting_name=pb_setting
      -setting_type=POWERBROKER
      -settings="SETTINGS,/etc/pbrun %RUNAS% %PROFILE% %command%
      ;PASSWORD_PROMPT_STRING,Password:"
      -separator=settings=";"
      -subseparator=settings=","
```

create_red_group

Defines a redundancy group: name and its members. After the redundancy group is created, you can edit the redundancy group from the Enterprise Manager Grid Control console to configure Charts to be displayed for redundancy group members.

Format

```
create_red_group
      -name="name"
      [-type=<generic_redundancy_group>]
      -add_targets="name1:type1;name2:type2;..."...
      [-owner=<Redundancy Group Owner>]
      [-timezone_region=<actual timezone region>]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the redundancy group.

- **type**

  Redundancy group type. Defaults to `generic_redundancy_group`.

- **add_targets**

  Add existing targets to the redundancy group. Each target is specified as a name-value pair `target_name:target_type`. The `-add_targets` option can be specified more than once.

- **owner**

  Owner of the redundancy group.

- **timezone_region**

  Time zone region of this redundancy group.

Examples

The following example creates a redundancy group named `lsnr_group`. This group consists of two Oracle listeners: `emp_rec` and `payroll`.

```
emcli create_red_group -name=lsnr_group
      -add_targets="emp_rec:oracle_listener"
      -add_targets="payroll:oracle_listener"
```

create_role

Creates a new Enterprise Manager admininistrator role.

Format

```
create_role
      -name="role_name"
      [-description="description"]
      [-roles="role1;role2;..."]
      [-users="user1;user2;..."]
      [-privilege="name;[[target_name:target_type]|jobid]"]...

[ ] denotes that the parameter is optional
```

Options

- **name**

  Role name.

- **description**

  Description of the role.

- **roles**

  List of roles to assign to this new role. Currently, the only built-in role is `PUBLIC`.

- **users**

  List of users to whom this role is assigned.

- **privilege**

  Privilege to grant to this role. This option can be specified more than once.
  Note: Privileges are case-insensitive.

  The following system privileges do not require a target or a job ID:

  – CREATE_ANY_ROLE

  – CREATE_ANY_PRIVILEGE

  – MANAGE_CREDENTIAL_GROUP

  – CREATE_TARGET

  – DELETE_ANY_TARGET

  – VIEW_ANY_TARGET

  – USE_ANY_BEACON

  – EM_MONITOR

  – SUPER_USER

  The following target privileges require specifying `target_name:target_type`:

  – VIEW_TARGET

  – OPERATOR_TARGET

  – MAINTAIN_TARGET

  – CLONE_FROM_TARGET

- FULL_TARGET

The following group privileges require specifying `target_name:target_type`:

- CREATE_TARGET_IN_GROUP

The following job privileges require specifying `jobid`:

- VIEW_JOB
- FULL_JOB

Examples

The following example creates a role named `my_new_role` with the one-sentence description - "This is a new role called `my_new_role`". The role combines three existing roles: `role1`, `role2`, and `role3`. The role also has two added privileges: to view the job with ID 923470234ABCDFE23018494753091111 and to view the target `host1.us.oracle.com:host`. The role is granted to `johndoe` and `janedoe`.

```
emcli create_role
      -name="my_new_role"
      -desc="This is a new role called my_new_role"
      -roles="role1;role2;role3"
      -privilege="view_job;923470234ABCDFE23018494753091111"
      -privilege="view_target;host1.us.oracle.com:host"
      -users="johndoe;janedoe"
```

create_service

Creates a service to be monitored by Enterprise Manager.

Format

```
create_service
      -name='name'
      -type='type'
      -availType=availability type (can be 'test' or 'system')
      -availOp=availability operator (can be 'and' or 'or')
      [-hostName=host name]
      [-agentURL=agent url]
      [-properties='pname1|pval1;pname2|pval2;...']
      [-timezone_region='gmt offset']
      [-systemname='system name']
      [-systemtype='system type']
      [-keycomponents='keycomp1name:keycomp1type;keycomp2name:keycomp2type;...']
      [-beacons='bcn1name:bcn1isKey;bcn2name:bcn2isKey;...']
      [-input_file='template:Template file name;[vars:Variables file name]']

[ ] denotes that the parameter is optional
```

Options

- **name**

  Service name. Names cannot contain colons (:), semi-colons (;), or any leading or trailing blanks.

- **type**

  Service type.

- **availType**

  Sets the availability to either test-based or system-based. If availability is set to `test`, template file, beacons, and variable are required arguments. If availability is set to `system`, systemname, systemtype, and keycomponents are required.

- **availOp**

  If `and`, it uses all key tests/components to decide availability. If `or`, it uses any key tests/components to decide availability.

- **hostName**

  Network name of the machine running the Management Agent that is collecting data for this target instance.

- **agentURL**

  URL of the Management Agent that is collecting data for this target instance. If host name is entered, the agent URL of the host is entered to this field automatically.

- **properties**

  Name-value pair (that is, prop_name|prop_value) list of properties for the service instance.

- **timezone_region**

  GMT offset for this target instance (-7 or -04:00 are acceptable formats).

- **systemname**

  System on which service resides.

- **keycomponents**

  Name-type pair (that is, `keycomp_name:keycomp_type`) list of key components in the system that are used for the service.

- **beacons**

  Name-isKey pairs that describe the beacons of the service. If isKey is set to `Y`, beacon is set as a key-beacon of the service. The service should have at least one key beacon if the availability is set to test-based.

- **input_file**

  Template file name is the XML file that includes the template definition. Variable file defines the values for the template.

Examples

The following example creates a generic service named `my service` with specified properties on a generic system named `my system` with specified key components. The availability is set as system-based.

```
emcli create_service
      -name='my service' -type='generic_service'
      -availType='system' -availOp='or'
      -properties='prop1:value1; prop2:value2'
      -timezone_region='PST8PDT'
      -systemname='my system' -systemtype='generic_system'
      -keycomponents='database:oracle_database; mytestbeacon:oracle_beacon'
```

create_system

Defines a system: name and its members. After the system is created, you can edit the system from the Enterprise Manager Grid Control console to configure charts to be displayed for system members.

Format

```
create_system
      -name="name"
      [-type=<system>]
      [-add_members="name1:type1;name2:type2;..."]...
      -timezone_region="actual timezone region"
      [-owner="owner"]
      [-meta_ver="meta version of system type"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the system.
- **type**

  System type: generic_system. Defaults to "generic_system".
- **add_members**

  Add existing targets to the system. Each target is specified as a name-value pair `target_name:target_type`. The `-add_members` option can be specified more than once.
- **timezone_region**

  Actual timezone region.
- **owner**

  Owner of the system.
- **meta_ver**

  Meta version of system type. Defaults to "1.0".

Examples

The following example creates a generic system named `db_system`. This system consists of two Oracle databases: `emp_rec` and `payroll`. The owner of this system is `user1`. The meta version of the system type is `3.0`.

```
emcli create_system -name=db_system
      -add_members="emp_rec:oracle_database"
      -add_members="payroll:oracle_database"
      -timezone_region="PST8PDT"
      -owner="user1"
      -meta_ver="3.0"
```

The following example creates a generic system named `my_system` that consists of an oracle database (`database2`), listener (`dblistener`), and host (`mymachine.myco.com`). The owner of this system is the logged-in user. The meta version of the system type is `1.0`.

```
emcli create_system -name=my_system
      -add_members="database2:oracle_database;dblistener:oracle_listener
      -add_members="mymachine.myco.com:host"
      -timezone_region="PST8PDT"
```

create_user

Creates a new Enterprise Manager administrator.

Format

```
create_user
      -name="name"
      -password="password"
      [-roles="role1;role2;..."]
      [-email="email1;email2;..."]
      [-privilege="name;[[target_name:target_type]|jobid]"]...

[ ] denotes that the parameter is optional
```

Options

- **name**

  Administrator name.

- **password**

  Administrator password.

- **roles**

  List of roles to grant to this administrator. Currently, the built-in roles include `PUBLIC`.

- **email**

  List of e-mail addresses for this administrator.

- **privilege**

  Privilege to grant to this administrator. You can specify this option more than once.

  The following system privileges do not require a target or a job ID:

  – `CREATE_ANY_ROLE`
  – `CREATE_ANY_PRIVILEGE`
  – `MANAGE_CREDENTIAL_GROUP`
  – `CREATE_TARGET`
  – `DELETE_ANY_TARGET`
  – `VIEW_ANY_TARGET`
  – `USE_ANY_BEACON`
  – `EM_MONITOR`
  – `SUPER_USER`

  The following target privileges require specifying `target_name:target_type`:

  – `VIEW_TARGET`
  – `OPERATOR_TARGET`
  – `MAINTAIN_TARGET`
  – `CLONE_FROM_TARGET`

– FULL_TARGET

The following group privileges require specifying target_name:target_type:

– CREATE_TARGET_IN_GROUP

The following job privileges require specifying jobid:

– VIEW_JOB
– FULL_JOB

Examples

The following example creates an Enterprise Manager administrator named new_admin. This administrator has two privileges: the ability to view the job with ID 923470234ABCDFE23018494753091111 and the ability to view the target host1.us.oracle.com:host. The administrator new_admin is granted the PUBLIC role.

```
emcli create_user
      -name="new_admin"
      -password="oracle"
      -email="first.last@oracle.com;joe.shmoe@shmoeshop.com"
      -roles="public"
      -privilege="view_job;923470234ABCDFE23018494753091111"
      -privilege="view_target;host1.us.oracle.com:host"
```

delete_blackout

Deletes a blackout that has already ended or has been fully stopped. You cannot delete a blackout that is either in progress or currently scheduled -- you need to first run `stop_blackout`.

Format

```
delete_blackout
      -name="name"
      [-createdby="blackout_creator" (default is current user)]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the blackout to delete.

- **createdby**

  Enterprise Manager user who created the blackout. The `SUPER_USER` privilege is required to delete a blackout created by another user.

Examples

The following example deletes blackout `backup_monthly` created by the current user.

```
emcli delete_blackout -name=backup_monthly
```

The following example deletes blackout `db_maintenance` that was created by Enterprise Manager administrator `sysadmin2`. The current user must either be user `sysadmin2` or a user with the `SUPER_USER` privilege.

```
emcli delete_blackout -name=db_maintenance -createdby=sysadmin2
```

delete_group

Deletes a group. Attempting to delete a non-existent group does not generate an error.

Format

```
delete_group
      -name="name"
      [-type=<group>]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of group to delete.

- **type**

  Group type: group. Defaults to "group".

Examples

The following example removes the group `payroll_group` that consists of database target types.

```
emcli delete_group -name=payroll_group
```

The following example removes the group `my_hosts` that consists of host target types.

```
emcli delete_group -name=my_hosts
```

The following example removes the group `my_group` that consists of mixed target types.

```
emcli delete_group -name=my_group
```

delete_job

Deletes a specified job. A job cannot be deleted if any of its executions are in the `EXECUTING` (Running) state. Use the `get_jobs` Verb to obtain a list of existing jobs along with their job IDs and statuses.

Format

```
delete_job
      -job_id="jobID" | -name="jobName"
```

Options

- job_id

  Job ID of the job to delete.

- name

  Name of the job to delete. To uniquely identify the job, the current user is used.

Examples

The following example deletes an existing job with the job ID 12345678901234567890123456789012.

```
emcli delete_job -job_id=12345678901234567890123456789012
```

The following example deletes an existing job named `my_job`, which belongs to the current Enterprise Manager user.

```
emcli delete_job -name=my_job
```

delete_metric_promotion

Deletes a promoted metric.

Format

```
delete_metric_promotion
      -name=Service target name
      -type=Service target type
      [-category = Usage/Performance]
      [-promotedMetricName = Promoted Metric]
      [-promotedMetricColumn = Promoted Metric Column]
      -promotedMetricKey = Key Value of the promoted metric

[ ] denotes that the parameter is optional
```

Options

- **category**

  Defines whether the promoted metric is a usage or a performance metric of a service. Category is used to determine the promoted metric name and metric column. If you do not specify this option, the `promotedMetricName` and `promotedMetricColumn` options must be specified.

- **promotedMetricName**

  Promoted metric name. This is optional if the category is specified.

- **promotedMetricColumn**

  Promoted metric column. This is optional if the category is specified.

- **promotedMetricKey**

  Required argument that determines the key value of the promoted metric. It is equivalent to the displayed name of the promoted metric in the UI.

Examples

The following example deletes the promoted Performance metric with key value `mymetric1` on service `MyTarget`.

```
emcli delete_metric_promotion -name='MyTarget' -type='generic_service'
      -category=Performance -promotedMetricKey=mymetric1
```

delete_privilege_delegation_settings

Deletes Sudo or PowerBroker settings.

Format

```
emcli delete_privilege_delegation_settings
                    -setting_names="setting_name1;setting_name2;setting_name3;"
```

Options

- **setting_names**

  Name of the settings you want to delete.

Example

The following example deletes the privilege settings for the names `setting_name1`, `setting_name2`, and `setting_name3`.

```
emcli delete_privilege_delegation_settings
        -setting_names="sudo_setting1;sudo_setting2;pbSetting1
```

delete_role

Deletes an existing Enterprise Manager administrator role.

Format

```
delete_role
      -name="role_name"
```

Options

- **name**

  Role name.

Examples

The following example deletes the role name `existing_role`.

```
emcli delete_role -name="existing_role"
```

delete_system

Deletes a system.

Format

```
delete_system
      -name="name"
      [-type=<generic_system>]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the system to delete.
- **type**

  System type: generic_system. Defaults to "generic_system".

Examples

The following example deletes the system `my_system`.

```
emcli delete_system -name=my_system
```

delete_target

Deletes a specified target from the Enterprise Manager Grid Control monitoring framework. Deleting a target removes it from the Management Repository and does not physically remove the target itself.

You can use the `get_targets` Verb to obtain a list of available targets and their respective types.

Format

```
delete_target
      -name="name"
      -type="type"
```

Options

- **name**

  Target name.

- **type**

  Target type.

Examples

The following example deletes the `oracle_database` target with the name `database`.

```
emcli delete_target
      -name="database"
      -type="oracle_database"
```

delete_test

Deletes a Services test along with its constituent steps and stepgroups.

Format

```
delete_test
      -name=target name
      -type=target type
      -testname=test name
      -testtype=test type
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **testname**

  Test name.

- **testtype**

  Test type.

Examples

The following example deletes the HTTP test named `MyTest` for the `generic_service` target named `MyTarget`.

```
emcli delete_test -name='MyTarget' -type='generic_service'
      -testname='MyTest' -testtype='HTTP'
```

delete_user

Deletes an existing Enterprise Manager administrator.

When a user is deleted, any jobs the user creates are stopped and deleted. Also, any blackouts the user creates are deleted. However, a user cannot be deleted if any blackouts created by the user are active at the time the call to delete the user is issued. This situation is considered an invalid state from which to delete a user. First, all of these active blackouts must be stopped, and a thwarted delete user call must be reissued.

Format

```
delete_user
      -name="user_name"
```

Options

- **name**

  Administrator name.

Examples

The following example deletes the Enterprise Manager administrator named `sysman3`.

```
emcli delete_user -name=sysman3
```

disable_test

Disables a Services test monitoring.

Format

```
disable_test
      -name=target name
      -type=target type
      -testname=test name
      -testtype=test type
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **testname**

  Test name.

- **testtype**

  Test type.

Examples

The following example disables the HTTP test named `MyTest` for the `generic_service` target named `MyTarget`.

```
emcli disable_test -name='MyTarget' -type='generic_service'
      -testname='MyTest' -testtype='HTTP'
```

enable_test

Enables a Services test monitoring. It pushes the Service test collection to all the beacons.

Format

```
enable_test
      -name=target name
      -type=target type
      -testname=test name
      -testtype=test type
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **testname**

  Test name.

- **testtype**

  Test type.

Examples

The following example enables the HTTP test named `MyTest` for the `generic_service` target named `MyTarget`.

```
emcli enable_test -name='MyTarget' -type='generic_service'
      -testname='MyTest' -testtype='HTTP'
```

execute_hostcmd

Executes a Host command across a set of targets.

Format

```
execute_hostcmd
      -cmd="host command"
      -osscript="os script to be executed with "cmd" "
      -targets="name1:type1;name2:type2;..."
      -credential_set_name="name"
      [-input_file="parameter_tag:script_file"]

[ ] denotes that the parameter is optional
```

Options

- **cmd**

  "host command" can be any valid host command or group of host commands.

- **targets**

  List of target-name, target-type pairs. The host command is executed across this list of Enterprise Manager targets. All targets must be of the type `host` or `composite`, which represents a group of targets. If it is a group, the group is expanded to extract all the host targets, and the host command is executed across these host targets.

- **credential_set_name**

  The `credential_set_name` parameter refers to the set name of the preferred credentials stored in the Enterprise Manager repository. If this parameter is not present, `HostCredsNormal` is used for executing host commands. For the `host` target type, two credential sets exist:

  – `HostCredsNormal` — Default unprivileged credential set for a host target

  – `HostCredsPriv` — Privileged credential set for a host target

    The credential set parameter can only be specified when the override credential parameters such as `username` and `password` are not present.

    If provided, the override credential parameters must be specified fully. For host command, `username` and `password` must be specified together.

- **input_file**

  Used in conjunction with the `-osscript` option, this option allows you to load the contents of an OS script. The `-input_file` option specifies a mapping between a tag and a local file path. The tag is specified in lieu of actual osscript contents of the `-osscript` option. The tag must not contain colons (:) or semi-colons (;).

Examples

The following example executes the host command `ls -l;` against the target `stach.us.oracle.com:host` and host targets contained in the group `grp`. The stored `HostCredsPriv` preferred credentials will be used for all the targets.

```
emcli execute_hostcmd
      -cmd="ls -l;"
```

```
      -credential_set_name="HostCredsPriv"
      -targets="stach.us.oracle.com:host;grp:composite"
```

The following example loads the contents of the script `/scratch/dba_scripts/shellscript.sh` into the value of option `-osscript` and executes it against target `reference.us.oracle.com:host` and host targets contained in the group `grp`. The stored `HostCredsNorma`' preferred credentials will be used for all the targets.

```
emcli execute_hostcmd
      -cmd="/bin/sh -s"
      -osscript="FILE"
      -input_file="FILE:/scratch/dba_scripts/shellscript.sh"
      -credential_set_name="HostCredsNormal"
      -targets="reference.us.oracle.com:host;grp:composite"
```

execute_sql

Executes a SQL command across a set of targets.

Format

```
execute_sql
      -sql="sql command"
      -targets="name1:type1;name2:type2;..."
      -credential_set_name="name"
      [-input_file="parameter_tag:script_file"]

[ ] denotes that the parameter is optional
```

Options

- **sql**

  "sql command" is a single SQL statement.

- **targets**

  List of target-name, target-type pairs. The SQL command executes across this list of Enterprise Manager targets. All targets must be of the type `oracle_database` or `composite`, which represents a group of targets. If it is a group, the group expands to extract all the database targets, and the SQL command is executed across these database targets.

- **credential_set_name**

  Refers to the set name of the preferred credentials stored in the Enterprise Manager repository. If this parameter is not present, the `DBCredsNormal` and `DBHostCreds` credential set is used for executing SQL commands. For each target type, several credential sets exist:

  – HostCredsNormal — Default unprivileged credential set for a host target

  – HostCredsPriv — Privileged credential set for a host target

  – DBHostCreds — Host credential set for an `oracle_database` target

  – DBCredsNormal — Default normal credential set for an oracle_database target

  – DBCredsSYSDBA — sysdba credential set for an `oracle_database` target

  You can only specify the `credential_set_name` parameter when the override credential parameters such as `[db_|host_]username` and `[db_|host_]password` are not present. If provided, the override credential parameters must be specified fully. For the SQL commands, `db_username`, `db_password`, `db_role`, `host_username`, and `host_password` must be present.

- **input_file**

  Used in conjunction with the `-sql` option, this option enables you to load the contents of a SQL script. The `-input_file` option specifies a mapping between a tag and a local file path. The tag is specified in lieu of an actual SQL command for the `-sql` option. The tag must not contain colons (:) or semi-colons (;).

Examples

The following example executes the SQL command `select * from sysman.mgmt_targets;` against the target `database:oracle_database` and database targets contained in the group grp. The stored SYSDBA preferred credentials will be used for all the targets.

```
emcli execute_sql
      -sql="select * from sysman.mgmt_targets;"
      -credential_set_name="DBCredsSYSDBA"
      -targets="database:oracle_database;grp:composite"
```

The following example loads the contents of the script /scratch/dba_scripts/enterprise_schema.sql into the value of option -sql, and executes it against target database:oracle_database and database targets contained in the group grp. The stored SYSDBA preferred credentials will be used for all the targets.

```
emcli execute_sql
      -sql="FILE"
      -input_file="FILE:/scratch/dba_scripts/enterprise_schema.sql"
      -credential_set_name="DBCredsSYSDBA"
      -targets="database:oracle_database;grp:composite"
```

export_template

Exports a monitoring template and also exports UDMs in the template. You can export a template to the file system in the form of an XML file, or you can print it on standard output in XML form.

Format

```
emcli export_template
      -name="name"
      -target_type="target_type"
      [-output_file=File to which template will be exported]

[ ] denotes that the parameter is optional
```

Options

- name

  Name of the template. The name and target type uniquely identify a template.

- **target_type**

  Target type of the template.

- output_file

  Specifies the file to output the template. If not specified, the template prints to `stdout`.

Examples

The following example shows that template XML specified by name `HOST_TEMP1` and target type `host` will be output to the screen.

```
emcli export_template -name=HOST_TEMP1 -target_type=host
```

The following example shows that template XML specified by name `HOST_TEMP1` and target type `host` will be created in the `test.xml` file.

```
emcli export_template -name=HOST_TEMP1 -target_type=host -output_file=test.xml
```

extend_as_home

Clones the specified Application Server Oracle Home or Software Library component from the target host to specified destinations. The new hosts join an existing cluster. For a Portal and Wireless install, OID user and password are also needed. For a J2EE instance connected to only a database-based repository, a DCM Schema password is needed.

Passing Variables Through EMCLI

When working with variables such as `%perlbin%` or `%oracle_home%`, EM CLI passes variable values from the current local environment instead of the variables themselves. To pass variables through an EM CLI command, as might be the case when using the `-prescripts` or `-postscripts` options, you can place the EM CLI command in a batch file and replace all occurrences of `%` with `%%`.

Format

```
emcli extend_as_home
       -input_file="dest_properties:file_path"
       -list_exclude_files="list of files to exclude"
       -isSwLib="true/false"
       -tryftp_copy="true/false"
       -jobname="name of cloning job"
       -iasInstance=instance
       -clustername=name of the cluster to join
       -oldIASAdminPassword=oldpass
       -newIASAdminPassword=newpass
       [-oiduser=oid admin user]
       [-oidpassword=oid admin password]
       [-dcmpassword=dcm schema password]
       [-prescripts=script name to execute"]
       [-run_prescripts_as_root="true/false"]
       [-postscripts=script to execute"]
       [-run_postscripts_as_root="true/false"]
       [-rootscripts=script name to execute"]
       [-swlib_component ="path:path to component;version:rev"]
       [-source_params="TargetName:name;HomeLoc:loc;HomeName:name;
         ScratchLoc:Scratch dir Location"
       [-jobdesc="description"]

[ ] denotes that the parameter is optional
```

Options

- **dest_properties**

  File containing information regarding the targets. Each line in the file corresponds to information regarding one destination.

  Format:

  ```
  Destination Host Name1;Destination Home Loc; Home Name;
  Scratch Location;
  ```

- **list_exclude_files**

  Comma-separated list of files to exclude. Not required if the source is a Software Library. You can use an asterisk "*" as a wildcard.

- **isSwLib**

Specifies whether it is an Oracle Home database or Software Library.

- **tryftp_copy**

  Try FTP to copy or not. You should set the FTP copy option to false when using EM CLI from the command line.

- **jobname**

  Name of the cloning job.

- **iasInstance**

  Application Server instance.

- **clustername**

  Name of the cluster to join.

- **oldIASAdminPassword**

  Old Application Server administrator password.

- **newIASAdminPassword**

  New Application Server administrator password.

- **oiduser**

  OID administrator user.

- **oidpassword**

  OID administrator password.

- **dcmpassword**

  DCM schema password.

- **prescripts**

  Path of the script to execute.

  Note: Double-quoted parameters can be passed using an escape (\) sequence. For example:

  ```
  prescripts=" <some value here>=\"some value here\" "
  ```

- **run_prescripts_as_root**

  Run prescripts as `root`. By default, this option is set to false.

- **postscripts**

  Path of the script to execute.

- **run_postscripts_as_root**

  Run postscripts as `root`. By default, this option is set to false.

- **rootscripts**

  Path of the script to execute. You can use the job system environment variables (`%oracle_home%`, `%perl_bin%`) to specify script locations.

- **swlib_component**

  Path to the Software Library to be cloned. `isSwLib` must be true in this case.

- **source_params**

Source Oracle home information. `isSwLib` must be false in this case.

- **jobdesc**

  Description of the job. If not specified, a default description is generated automatically.

extend_crs_home

Extends an Oracle Clusterware cluster, using an Oracle Clusterware source home location or an Oracle Clusterware Software Library component, to specified destinations. If a component is used, you must provide information for a host that is part of the current cluster, along with the Oracle Home name and home location. When cloning from a source home, you do not need to pass source information twice (`-srchost`, `-home_name`, and `-home_location`). This information is extracted from the home. These are only needed when cloning from a Software Library component.

Format

```
emcli extend_crs_home
   -input_file="dest_properties:file_path"
   -list_exclude_files="list of files to exclude"
   -clusternodes="node1;node2;node3;node4"
   -isSwLib="true/false"
   -tryftp_copy="true/false"
   -jobname="name of cloning job"
   -ocrLoc=ocr location
   -vdiskLoc=voting disk location
    [-srchost=name of a host node present on the cluster being extended"]
    [-home_name="home name on a host for the existing Oracle Clusterware
      cluster"]
    [-home_location="location on a host for the existing Oracle Clusterware
      cluster"]
    [-prescripts=script name to execute"]
    [-run_prescripts_as_root="true/false"]
    [-postscripts=script to execute"]
    [-run_postscripts_as_root="true/false"]
    [-rootscripts=script name to execute"]
    [-swlib_component ="path:path to component;version:rev"]
    [-source_params="TargetName:name;HomeLoc:loc;HomeName:name;
      ScratchLoc:Scratch dir Location"]
    [-jobdesc="description"]

[ ] denotes that the parameter is optional
```

Options

- **dest_properties**

  File containing information regarding the targets. Each line in the file corresponds to information regarding one destination.

  Format:

  `Destination Host Name1;Destination Node Name;Scratch Location;PVTIC;VirtualIP;`

- **list_exclude_files**

  Comma-separated list of files to exclude. Not required if the source is a Software Library. You can use an asterisk "*" as a wildcard.

- **clusternodes**

  List of current nodes in the cluster.

- **isSwLib**

Specifies whether it is an Oracle Home database or Software Library.

- **tryftp_copy**

  Try FTP to copy or not. You should set the FTP copy option to false when using EM CLI from the command line.

- **jobname**

  Name of the Cloning job.

- **srchost**

  Name of a host that is part of the Oracle Clusterware cluster being extended.

- **home_name**

  Name of home used by all the current Oracle Clusterware cluster nodes.

- **home_location**

  Home location used by all the current Oracle Clusterware cluster nodes.

- **ocrLoc**

  Oracle Cluster Registry Location.

- **vdiskLoc**

  Voting disk location.

- **prescripts**

  Path of the script to execute.

  > **Note:** Double-quoted parameters can be passed using an escape (\) sequence. For example:
  >
  > ```
  > prescripts=" <some value here>=\"some value here\" "
  > ```

- **run_prescripts_as_root**

  Run prescripts as `root`. By default, this option is set to false.

- **postscripts**

  Path of the script to execute.

- **run_postscripts_as_root**

  Run postscripts as `root`. By default, this option is set to false.

- **rootscripts**

  Path of the script to execute. You can use the job system environment variables (`%oracle_home%`, `%perl_bin%`) to specify script locations.

- **swlib_component**

  Path to the Software Library to be cloned. `isSwLib` must be true in this case.

- **source_params**

  Source Oracle home info. `isSwLib` must be false in this case.

- **jobdesc**

  Description of the job. If not specified, a default description is generated automatically.

Examples

```
emcli extend_crs_home -input_file="dest_properties:crs.prop"  -list_exclude_
files=""
   -isSwLib="false"
   -tryftp_copy="false" -jobname="crs extend job"
   -home_name="cloneCRS1"
   -home_location="/scratch/scott/cloneCRS1 "
   -clusternodes="node1;node2" -clustername="crscluster"
   -postscripts="%perlbin%/perl%emd_root%/admin/scripts/cloning/samples/post_crs_
extend.pl ORACLE_HOME=%oracle_home%"
   -run_postscripts_as_root="false" -rootscripts="%oracle_home%/root.sh"
   -source
params="TargetName:testhost;HomeLoc:/scratch/scott/cloneCRS1;HomeName:cloneCRS1;Sc
ratchLoc:/tmp"
```

Passing Variables Through EMCLI

When working with variables such as `%perlbin%` or `%oracle_home%`, EM CLI passes variable values from the current local environment instead of the variables themselves. To pass variables through an EM CLI command, as might be the case when using the `-prescripts` or `-postscripts` options, you can place the EM CLI command in a batch file and replace all occurrences of `%` with `%%`.

extend_rac_home

Extends a RAC cluster by cloning a specified Oracle Home location or a RAC Software Library component to specified destinations. If a component is used, you must provide information for a host that is part of the current cluster, along with the Oracle Home name and home location. When cloning from a source home, this information is automatically extracted from the home.

Format

```
emcli extend_rac_home
   -input_file="dest_properties:file_path"
   -list_exclude_files="list of files to exclude"
   -isSwLib="true/false"
   -tryftp_copy="true/false"
   -jobname="name of cloning job"
   -clusternodes="node1;node2;node3;node4"
    [-srchost=name of a host node present on the RAC cluster being extended"]
    [-home_name="home name on a host for the existing RAC cluster"]
    [-home_location="location on a host for the existing RAC cluster"]
    [-prescripts="script name to execute"]
    [-run_prescripts_as_root="true/false"]
    [-postscripts="script to execute"]
    [-run_postscripts_as_root="true/false"]
    [-rootscripts="script name to execute"]
    [-swlib_component ="path:path to component;version:rev"]
    [-source_params="TargetName:name;HomeLoc:loc;HomeName:name;
      ScratchLoc:Scratch dir Location"]
    [-jobdesc="description"]

[ ] denotes that the parameter is optional
```

Options

- **dest_properties**

  File containing information regarding the targets. Each line in the file corresponds to information regarding one destination.

  Format:

  `Destination Host Name;Destination Node Name;Scratch Location;`

- **list_exclude_files**

  Comma-separated list of files to exclude. Not required if the source is a Software Library. You can use an asterisk "*" as a wildcard.

- **isSwLib**

  Specifies whether it is an Oracle Home database or Software Library.

- **tryftp_copy**

  Try FTP to copy or not. You should set the FTP copy option to false when using EM CLI from the command line.

- **jobname**

  Name of the cloning job.

- **clusternodes**

Current nodes in the cluster.

- **srchost**

  Name of a host that is part of the RAC cluster being extended.

- **home_name**

  Name of the home used by all the current RAC cluster nodes.

- **home_location**

  Home location used by all the current RAC cluster nodes.

- **prescripts**

  Path of the script to execute.

  > **Note:** Double-quoted parameters can be passed using an escape (\) sequence. For example:
  >
  > ```
  > prescripts=" <some value here>=\"some value here\" "
  > ```

- **run_prescripts_as_root**

  Run prescripts as `root`. By default, this option is set to false.

- **postscripts**

  Path of the script to execute.

- **run_postscripts_as_root**

  Run postscripts as `root`. By default, this option is set to false.

- **rootscripts**

  Path of the script to execute.

- **swlib_component**

  Path to the Software Library being cloned. `isSwLib` must be true in this case.

- **source_params**

  Source Oracle home info. `isSwLib` must be false in this case.

- **jobdesc**

  Description of the job. If not specified, a default description is generated automatically.

Examples

```
emcli extend_rac_home
      -input_file="dest_properties:clonedestinations"
      -list_exclude_files="*.log,*.dbf,sqlnet.ora,tnsnames.ora,listener.ora"
      -isSwLib="false"
      -tryftp_copy="false"
      -jobname="clone database home"
      -clusternodes="node1;node2"
      -prescripts="/home/joe/myScript"
      -run_prescripts_as_root="true"
      -rootscripts="%oracle_home%/root.sh"
      -source_params="TargetName:host.domain.com;HomeLoc:/oracle/database1;
       HomeName:OUIHome1;ScratchLoc:/tmp"
```

Passing Variables Through EMCLI

When working with variables such as `%perlbin%` or `%oracle_home%`, EM CLI passes variable values from the current local environment instead of the variables themselves. To pass variables through an EM CLI command, as might be the case when using the `-prescripts` or `-postscripts` options, you can place the EM CLI command in a batch file and replace all occurrences of `%` with `%%`.

extract_template_tests

Extracts variables and test definitions from a repository template into a local file.

Format

```
extract_template_tests
      -templateName=<template name>
      -templateType=<template type>
      -output_file=<output filename>
      [-encryption_key=<key>]

[ ] denotes that the parameter is optional
```

Options

- **templateName**

  Template name.

- **templateType**

  Template type.

- **output_file**

  Name of the output file. If the file does not exist, it will be created; if it already exists, it will be overwritten. (This is assuming the extract operation was successful; if the operation fails, no files are created, and any existing files are left unchanged.)

- **encryption_key**

  Optional key to encrypt the file contents. The same key should be used to decrypt the file.

Examples

The following example creates a file named `my_template.xml` containing the variable values and test definitions of the Web Application template `my_template`. The file contents are encrypted using the key `my_password`.

```
emcli extract_template_tests
      -templateName='my_template' -templateType='website'
      -output_file='my_template.xml' -encryption_key='my_password'
```

get_aggregate_service_info

Gets timezone and availability evaluation function information of an aggregates service instance.

Format

```
get_aggregate_service_info
      -name="name"
      -type="type"
      [-noheader]
      [-script|-format=
      [name:"pretty|script|csv"];
      [column_separator:"sep_string"];
      [row_separator:"row_sep_string"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Aggregate service name.

- **type**

  Aggregate service type.

Examples

```
emcli get_aggregate_service_info -name="My_Name"
     -type="aggregate_service"
```

get_aggregate_service_members

Gets sub-services of an aggregate service instance.

Format

```
get_aggregate_service_members
      -name="name"
      -type="type"
      [-noheader]
      [-script|-format=
      [name:"pretty|script|csv"];
      [column_separator:"sep_string"];
      [row_separator:"row_sep_string"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Aggregate service name.

- **type**

  Aggregate service type.

Examples

```
emcli get_aggregate_service_members -name="My_Name"
      -type="aggregate_service"
```

get_blackout_details

Gets detailed information for a specified blackout.

Format

```
get_blackout_details
      -name="name"
      [-createdby="blackout_creator" (default is current user)]
      [-noheader]
      [-script | -format=
      [name:<pretty|script|csv>];
      [column_separator:"column_sep_string"];
      [row_separator:"row_sep_string"];
      ]
[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the blackout.

- **createdby**

  Enterprise Manager user who created the blackout.

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Status, Status ID, Run Jobs, Next Start, Duration, Reason, Frequency, Repeat, Days, Months, Start Time, End Time, TZ Offset

Examples

The following example shows detailed information for blackout `blackout1` that the current user created.

```
emcli get_blackout_details -name=blackout1
```

The following example shows detailed information for blackout `blackout1` that user `joe` created.

```
emcli get_blackout_details -name=blackout1 -createdby=joe
```

get_blackout_reasons

Lists all blackout reasons, one per line.

Format

```
get_blackout_reasons
```

Examples

The following example lists all blackout reasons, one per line.

```
emcli get_blackout_reasons
```

get_blackout_targets

Lists targets for a specified blackout.

Format

```
get_blackout_targets
      -name="name"
      [-createdby="blackout_creator" (default is current user)]
      [-noheader]
      [-script | -format=
      [name:<pretty|script|csv>];
         [column_separator:"column_sep_string"];
          [row_separator:"row_sep_string"];
      ]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the blackout.

- **createdby**

  Enterprise Manager user who created the blackout.

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Target Name, Target Type, Status, Status ID

Examples

The following example lists targets in the blackout `blackout1` the current user created.

```
emcli get_blackout_targets -name=blackout1
```

The following example lists targets in the blackout `blackout1` that user `joe` created.

```
emcli get_blackout_targets -name=blackout1 -createdby=joe
```

get_blackouts

Lists all blackouts or just those for a specified target or one or more hosts. Only the blackouts the user has privilege to view are listed.

Format

```
get_blackouts
      [-target="name1:type1" | -hostnames="host1;host2;..."]
      [-noheader]
      [-script | -format=
      [name:<pretty|script|csv>];
         [column_separator:"column_sep_string"];
         [row_separator:"row_sep_string"];
      ]

[ ] denotes that the parameter is optional
```

Options

- **target**

  List blackouts for this target. When neither this option nor the `-hostnames` option is specified, all blackouts the user has privilege to view are listed.

- **hostnames**

  List blackouts which have a target on one of the specified hosts. The hostname is just the target name part of the host target. For example, specify `host.us.oracle.com`, rather than `host.us.oracle.com:host`. When neither this option nor the `-target` option is specified, all blackouts the user has privilege to view are listed.

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Column

Name, Created By, Status, Status ID, Next Start, Duration, Reason, Frequency, Repeat, Start Time, End Time, Previous End, TZ Offset

Examples

The following example shows all blackouts with some details.

```
emcli get_blackouts
```

The following example shows all blackouts that cover the target `database2:oracle_database`.

```
emcli get_blackouts -target=database2:oracle_database
```

The following example shows all blackouts that cover some target on host `myhost.us.oracle.com`.

```
emcli get_blackouts -hostnames=myhost.us.oracle.com
```

The following example shows all blackouts that cover some target on host `myhost.us.oracle.com` or on host `yourhost.us.oracle.com`.

```
emcli get_blackouts -hostnames="myhost.us.oracle.com"
-hostnames="yourhost.us.oracle.com"
```

get_group_members

Lists the members of the specified group.

Note that targets are only listed once, even though they can be in more than one sub-group of the group.

Format

```
get_group_members
      -name="name"
      [-type=<group>]
      [-depth=#  (default 1)]
      [-noheader]
      [-script | -format=
         [name:<pretty|script|csv>];
         [column_separator:"column_sep_string"];
         [row_separator:"row_sep_string"];
      ]
[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name of the group.

- **type**

  Group type: group. Defaults to `group`.

- **depth**

  List target members in sub-groups to the depth specified. When the depth is set to 0, no group target members are listed, and only the group's existence is verified. When the depth is set to -1, all group and sub-group target members are listed; in this case no groups will appear in the output. Note that a target is listed at most once, even though it can be a member of several sub-groups.

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Target Name, Target Type

Examples

The following example lists the databases in group db2_group.

```
emcli get_group_members -name=db2_group
```

The following example verifies that group my_hosts:group exists.

```
emcli get_group_members -name=my_hosts -depth=0
```

The following example lists the unique targets in group my_group:group and its sub-groups.

```
emcli get_group_members -name=my_group -depth=-1
```

get_groups

Lists all groups.

Format

```
get_groups
      [-noheader]
      [-script | -format=
         [name:<pretty|script|csv>];
         [column_separator:"column_sep_string"];
         [row_separator:"row_sep_string"];
      ]

[ ] denotes that the parameter is optional
```

Options

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Target Name, Target Type

Examples

The following example lists all groups.

```
emcli get_groups
```

get_instance_data_xml

Downloads instance data XML and generate an XML file containing that data.

Format

```
emcli get_instance_data_xml -instance={instance_guid}
```

Options

- **instance**

  Instance GUID.

Examples

```
emcli get_instance_data_xml -instance=16B15CB29C3F9E6CE040578C96093F61 > data.xml
```

get_instances

Displays a list of procedure instances.

See Also: `get_procedure_types`

Format

```
emcli get_instances -type={procedure type}
```

Options

- **type={procedure type}**

  Display all the Procedure Instances of type {procedure type}.

Output Columns

GUID, Procedure Type, Instance Name, Status

get_jobs

Lists existing jobs.

Format

```
get_jobs
      [-job_ids="ID1;ID2;..."]
      [-targets="type1:name1;type2:name2;..."]
      [-status_ids="status1;status2;..."]
      [-noheader]
      [-script | -format=
            [name:<pretty|script|csv>];
            [column_separator:"column_sep_string"];
            [row_separator:"row_sep_string"];
      ]

[ ] denotes that the parameter is optional
```

Options

- **job_ids**

  List of job IDs to use as the output filters.

- **targets**

  List of targets (as name-type pairs) to use as the output filters.

- **status_ids**

  List of numeric status IDs to use as the output filters.

  The numeric codes for all possible job statuses are as follows:

  – SCHEDULED=1
  – EXECUTING(Running)=2
  – ABORTED(Failed Initialization)=3
  – FAILED=4
  – COMPLETED(Successful)=5
  – SUSPENDED_USER=6
  – SUSPENDED_AGENT_DOWN=7
  – STOPPED=8
  – SUSPENDED_LOCK=9
  – SUSPENDED_EVENT=10
  – SUSPENDED_BLACKOUT=11
  – STOP_PENDING=12
  – SUSPEND_PENDING=13
  – INACTIVE=14
  – QUEUED=15

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Name, Type, ID, Execution ID, Scheduled, Completed, Status, Status ID, Owner, Target Type, Target Name

Examples

The following example shows the jobs with the specified job IDs.

```
emcli get_jobs
      -job_ids="12345678901234567890123456789012,
09876543210987654321098765432100"
```

The following example shows all jobs run against a host target named `mainhost.us.oracle.com` that are scheduled or have completed.

12345678901234567890123456789012 and 09876543210987654321098765432100.

```
emcli get_jobs
      -status_ids="1,5"
      -targets="mainhost.us.oracle.com:host"
```

The following example shows all jobs run against an Oracle database target named `payroll` that have failed. Tabular output is generated using tabs as column separators and newlines as row separators.

```
emcli get_jobs
      -status_ids=4
      -targets="payroll:oracle_database"
      -script
```

get_procedure_types

Gets the list of all Deployment Procedure types.

Format

```
emcli get_procedure_types
```

Output Column

Procedure Type

get_procedure_xml

Gets the Deployment Procedure XML file. XML will be printed on standard output.

Format

```
emcli get_procedure_xml -procedure={procedure guid}
```

Options

- **procedure**

  Procedure GUID.

Output

The Deployment Procedure XML.

Examples

```
emcli get_procedure_xml -procedure=16B15CB29C3F9E6CE040578C96093F61 > proc.xml
```

get_procedures

Gets a list of Deployment Procedures.

See Also: `get_procedure_types`

Format

```
emcli get_procedures [-type={procedure type}]

[ ] denotes that the parameter is optional
```

Options

- **-type={procedure type}**

  Display all the Deployment Procedures of type {procedure type}.

Output Columns

GUID, Procedure Type, Name, Version, Created By

get_system_members

Lists the members of the specified system.

Format

```
get_system_members
      -name="name"
      [-type=<generic_system>]
      [-depth=#  (default 1)]
      [-noheader]
      [-script | -format=
      [name:<pretty|script|csv>];
            [column_separator:"column_sep_string"];
            [row_separator:"row_sep_string"];
      ]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name of the system.

- **type**

  System type: `generic_system`. Defaults to `generic_system`.

- **depth**

  List target members in sub-systems to the depth specified. When the depth is set to 0, no system target members are listed, and only the system's existence is verified. When the depth is set to -1, all system and sub-system target members are listed.

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Source Target Name, Member Target Name, Member Target Type, Level

Examples

The following example lists the databases in system `db2_system`.

```
emcli get_system_members -name=db2_system
```

The following example verifies that system `my_system:generic_system` exists.

```
emcli get_system_members -name=my_system -depth=0
```

The following example lists the unique targets in `system my_system:generic_system` and its sub-systems.

```
emcli get_system_members -name=my_system -depth=-1
```

get_targets

Gets status and alert information for targets.

Format

```
get_targets
      [-targets="[name1:]type1;[name2:]type2;..."]
      [-alerts]
      [-noheader]
      [-script | -format=
      [name:<pretty|script|csv>];
            [column_separator:"column_sep_string"];
            [row_separator:"row_sep_string"];
      ]

[ ] denotes that the parameter is optional
```

Options

- **targets=name:type**

  Name or type can be either a full value or a pattern match using `%`. Also, name is optional, so the type can be specified alone.

- **alerts**

  Shows the count of critical and warning alerts for each target.

- **noheader**

  Display tabular output without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Status ID, Status, Target Type, Target Name, Critical, Warning

Examples

The following example shows all targets. Critical and Warning columns are not included.

```
emcli get_targets
```

The following example shows all targets. Critical and Warning columns are shown.

```
emcli get_targets
      -alerts
```

The following example shows all `oracle_database` targets.

```
emcli get_targets
      -targets="oracle_database"
```

The following example shows all targets whose type contains the string `oracle`.

```
emcli get_targets
      -targets="%oracle%"
```

The following example shows all targets whose name starts with `databa` and type contains `oracle`.

```
emcli get_targets
      -targets="databa%:%oracle%"
```

The following example shows status and alert information on the Oracle database named `database3`.

```
emcli get_targets
      -targets="database3:oracle_database"
      -alerts
```

grant_privs

Grants the privileges to the existing Enterprise Manager user or Enterprise Manager Role.

Format

```
emcli grant_privs -name="username/rolename"
     [-privilege="name;[[target_name:target_type]|jobid]"]...

[ ] denotes that the parameter is optional
```

Options

- **name**

  User name or role name to which privileges will be assigned.

- **privilege**

  Privilege, which will be granted to the Enterprise Manager user or role. This option can be specified more than once.

  The following system privileges do not require a target or a job ID:

  CREATE_ANY_ROLE

  CREATE_TARGET

  DELETE_ANY_TARGET

  VIEW_ANY_TARGET

  USE_ANY_BEACON

  EM_MONITOR

  The following target privileges require specifying `target_name:target_type`:

  VIEW_TARGET

  OPERATOR_TARGET

  FULL_TARGET

  The following group privileges require specifying `target_name:target_type`:

  CREATE_TARGET_IN_GROUP

  The following job privileges require specifying `jobid`:

  VIEW_JOB

  FULL_JOB

Examples

```
1. emcli grant_privs
          -name="user1"
          -privilege="USE_ANY_BEACON"
          -privilege="FULL_JOB;923470234ABCDFE23018494753091111"
          -privilege="FULL_TARGET;host1.us.oracle.com:host"
Grant the privileges to Enterprise Manager user : user1
Three privileges are grant to user1 :
    1. Privilege to use any beacon
    2. Full control on the Jobs with ID 923470234ABCDFE23018494753091111
```

```
    3. Full control on the target host1.us.oracle.com:host

2. emcli grant_privs
          -name="Role1"
          -privilege="FULL_TARGET;host1.us.oracle.com:host"
Grant the target privileges to EM Role  : Role1
```

grant_roles

Grants the roles to an existing Enterprise Manager user or Enterprise Manager role.

Format

```
emcli grant_roles -name="username/rolename"
     [-roles="role1;role2;..."]

[ ] denotes that the parameter is optional
```

Options

- **name**

  User name or role name to which roles will be assigned.

- **roles**

  Roles that will be granted to an Enterprise Manager user or role. You can specify this option more than once.

Examples

```
1. emcli grant_roles
        -name="user1"
        -roles="SUPER_USER"
2. emcli grant_roles
        -name="Role1"
        -roles="BLACKOUT_ADMIN;MAINTAIN_TARGET"
```

import_template

Imports a monitoring template from an XML file. The resulting definition is saved in the repository.

Format

```
emcli import_template
      -files="file1;file2;..."
```

Options

- files

  Path/file name of an XML file, which contains a valid template definition. You can specify multiple files this option by separating each file with a semi-colon (`;`).

Examples

The following example imports a template from `template.xml` file.

```
emcli import_template -files="template.xml"
```

The following example imports three templates — one from each of the files specified.

```
emcli import_template -files="e1.xml;e2.xml;e3.xml"
```

help

Shows a summary of all verbs or command line help for individual EM CLI verbs.

> **Note:** EM CLI must be set up and configured before command line help is available for all verbs.

Format

```
help [verbname]
help

[ ] denotes that the parameter is optional
```

Options

None.

Examples

```
emcli help add_target (provides description, syntax, and usage examples for a
specific Verb.)
emcli help (provides an overview for all available verbs.)
```

modify_aggregate_service

Modifies an aggregate service instance.

Format

```
modify_aggregate_service
      -name="name"
      -type="type"
      [-add_sub_services="name1:type1;name2:type2;..."]
      [-del_sub_services="name1:type1;name2:type2;..."]
      [-avail_eval_func="function to evaluate availability."]
      [-timezone_region="timezone region"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Aggregate service name.

- **type**

  Aggregate service type.

- **add_sub_services**

  Sub-services to be added.

- **del_sub_services**

  Sub-services to be deleted.

- **avail_eval_func**

  PL/SQL function to evaluate the availability of the aggregate service. Use [or|and] for predefined evaluation helper function.

- **timezone_region**

  Time Zone Region of the service.

Examples

```
emcli modify_aggregate_service -name="My_Name"
      -type="aggregate_service"
      -add_sub_services="sub1:type1;sub2:type2"
      -del_sub_services="sub3:type3"
      -avail_eval_func="my_pkg.my_eval_func"
      -timezone_region="CST"
```

modify_group

Adds or removes targets from an existing group.

An error is not generated when attempting to delete a non-existent target in the group or when attempting to add a target that already exists in the group.

Format

```
modify_group
      -name="name"
      [-type=<group>]
      [-add_targets="name1:type1;name2:type2;..."]...
      [-delete_targets="name1:type1;name2:type2;..."]...

[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name of the group to modify.

- **type**

  Group type: group. Defaults to `group`.

- **add_targets**

  Targets to add, each specified as `target_name:target_type`. Option `-add_targets` can be specified more than once.

- **delete_targets**

  Targets to delete, each specified as `target_name:target_type`. Option `-delete_targets` can be specified more than once.

Examples

The following example modifies group `db2_group` by adding database `database:oracle_database` and deleting database `database2:oracle_database` from the group.

```
emcli modify_group -name=db2_group
      -add_targets=database:oracle_database
      -delete_targets=database2:oracle_database
```

The following example modifies group `my_hosts` by adding host `yourhost.us.oracle.com:host` to the group.

```
 emcli modify_group -name=my_hosts
      -add_targets=yourhost.us.oracle.com:host
```

The following example modifies group `my_group` by adding targets `group_a:group` and `database:oracle_database` and deleting the nonexistent target `nogroup:group` from the group.

```
emcli modify_group -name=my_group
      -add_targets=group_a:group
      -add_targets=database:oracle_database
      -delete_targets=nogroup:group
```

modify_red_group

Adds or removes targets from an existing redundancy group.

An error is not generated when attempting to delete a non-existent target in the redundancy group.

Format

```
modify_red_group
      -name="name"
      -type=<generic_redundancy_group>
      [-add_targets="name1:type1;name2:type2;..."]...
      [-delete_targets="name1:type1;name2:type2;..."]...
      [-owner=<Redundancy Group Owner>]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name of the group to modify.

- **type**

  Redundancy Group type: generic_redundancy_group. Defaults to `generic_redundnacy_group`.

- **add_targets**

  Targets to add, each specified as `target_name:target_type`. Option `-add_targets` can be specified more than once.

- **delete_targets**

  Targets to delete, each specified as `target_name:target_type`. Option `-delete_targets` can be specified more than once.

- **owner**

  Owner of the redundancy group.

Examples

The following example modifies redundancy group Servers by adding Oracle Apache `Server1:oracle_apache` and deleting Oracle Apache `Server5:oracle_apache` from the redundancy group.

```
emcli modify_red_group -name=Servers
      -add_targets=HTTP_Server1:oracle_apache
      -delete_targets=Server5:oracle_apache
```

modify_role

Modifies an existing Enterprise Manager administrator role.

Note: Omit an argument to leave its value unchanged.

Format

```
modify_role
       -name="role_name"
       [-description="description"]
       [-roles="role1;role2;..."]
       [-privilege="name;[[target_name:target_type]|jobid]"]...
       [-users="user1;user2;..."]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Role name.

- **description**

  Replace the description of the role.

- **roles**

  Replace the list of roles assigned to this existing role. Currently, the only built-in role is PUBLIC.

- **users**

  Replace the list of users to whom this role is assigned.

- **privilege**

  Replace privileges granted to this role. This option can be specified more than once.

  Note: Privileges are case-insensitive.

  The following system privileges do not require a target or a job ID:

  – `CREATE_ANY_ROLE`

  – `CREATE_ANY_PRIVILEGE`

  – `MANAGE_CREDENTIAL_GROUP`

  – `CREATE_TARGET`

  – `DELETE_ANY_TARGET`

  – `VIEW_ANY_TARGET`

  – `USE_ANY_BEACON`

  – `EM_MONITOR`

  – `SUPER_USER`

  The following target privileges require specifying `target_name:target_type`:

  – `VIEW_TARGET`

  – `OPERATOR_TARGET`

- MAINTAIN_TARGET
- CLONE_FROM_TARGET
- FULL_TARGET

The following group privileges require specifying `target_name:target_type`:

- CREATE_TARGET_IN_GROUP

The following job privileges require specifying `jobid`:

- VIEW_JOB
- FULL_JOB

Examples

The following example modifies a role named `existing_role` with the one-sentence description `This role was changed`. The role combines three existing roles: `role1`, `role2`, and `role3`. The role also has two added privileges: to view the job with ID `923470234ABCDFE23018494753091111` and to view the target `host1.us.oracle.com:host`. The role is granted to `johndoe` and `janedoe`.

```
emcli modify_role
      -name="existing_role"
      -desc="This role was changed"
      -roles="role1;role2;role3"
      -privilege="view_job;923470234ABCDFE23018494753091111"
      -privilege="view_target;host1.us.oracle.com:host"
      -users="johndoe;janedoe"
```

The following example modifies a role named `existing_role` by assigning `role4`, `role5`, and `role6` to it. The description, privileges, and users associated with this role remain unchanged.

```
emcli modify_role
      -name="existing_role"
      -roles="role4;role5;role6"
```

modify_system

Adds or removes targets from an existing system.

An error is not generated when attempting to delete a non-existent target in the system or when attempting to add a target that already exists in the system.

If you specify both the -add_members and -delete_members options in the same command, the members specified by -delete_members are deleted first, then the members specified by -add_members are added.

Format

```
modify_system
      -name="name"
      [-type=<generic_system>]
      [-add_members="name1:type1;name2:type2;..."]...
      [-delete_members="name1:type1;name2:type2;..."]...
      [-owner="new_owner"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name of the system to modify.

- **type**

  System type: generic_system. Defaults to generic_system.

- **add_members**

  Targets to add, each specified as target_name:target_type. You can specify the option -add_members more than once.

- **delete_members**

  Targets to delete, each specified as target_name:target_type. Option -delete_members can be specified more than once.

- **owner**

  New owner of the system.

Examples

The following example modifies system db2_system by adding database database:oracle_database and deleting database database2:oracle_database from the system. The new owner of the system is user2.

```
emcli modify_system -name=db2_system
      -add_members=database:oracle_database
      -delete_members=database2:oracle_database
      -owner=user2
```

The following example modifies system my_hosts by adding host yourhost.us.oracle.com:host to the system.

```
emcli modify_system -name=my_hosts
      -add_members=yourhost.us.oracle.com:host
```

The following example modifies system `my_system` by adding targets `system_a:generic_system` and `database:oracle_database`, and deleting the nonexistent target `nosystem:generic_system` from the system.

```
emcli modify_system -name=my_system
      -add_members=system_a:generic_system
      -add_members=database:oracle_database
      -delete_members=nosystem:generic_system
```

modify_target

Modifies a target instance definition.

Format

```
modify_target
      -name="name"
      -type="type"
      [-properties="pname1:pval1;pname2:pval2;..."]...
      [-separator=properties="sep_string"]
      [-subseparator=properties="subsep_string"]
      [-credentials="userpropname:username;pwdpropname:password;..."]
      [-input_file="parameter_tag:file_path"]
      [-display_name="display name"]
      [-on_agent]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name.

- **type**

  Target type.

- **properties**

  Name-value pair list of properties for the target instance. The "name"(s) are identified in the target-type metadata definition. They must appear exactly as they are defined in that file. Metadata files are located in `$AGENT_ORACLE_HOME/sysman/admin/metadata`.

- **separator=properties**

  Specify a string delimiter to use between name-value pairs for the value of the `-properties` option. The default separator delimiter is ";".

- **subseparator=properties**

  Specify a string delimiter to use between name and value in each name-value pair for the value of the `-properties` option. The default subseparator delimiter is ":".

- **credentials**

  Monitoring credentials (name-value pairs) for the target instance. The "name"(s) are identified in the target-type metadata definition as credential properties. They must appear exactly as they are defined in that file. Metadata files are located in `$AGENT_ORACLE_HOME/sysman/admin/metadata`.

- **input_file**

  Used in conjunction with the `-credentials` option, this option enables you to store specific target monitoring credential values, such as passwords, in a separate file. The `-input_file` option specifies a mapping between a tag and a local file path. The tag is specified in lieu of specific monitoring credentials of the `-credentials` option. The tag must not contain colons (:)or semi-colons (;)

- **display_name**

Set target display name.

- **on_agent**

  Propagates changes to the Management Agent collecting this target's metrics.

Examples

The following example modifies the display name to `New Name DB` for the database with the internal name `database`.

```
emcli modify_target
      -name="database"
      -type="oracle_database"
      -display_name="New Name DB"
```

The following example modifies the credentials for the `oracle_database` target with the name `database`. This example illustrates the use of the `input_file` to camouflage the credentials. The password is actually in a file named `at_pwd_file`. The `input_file` argument replaces `PWD_FILE` with the contents of the `at_pwd_file` in the `credentials` argument. The `on_agent` flag ensures that the changes are propagated to the Management Agent collecting for this target.

```
emcli modify_target
      -name="database"
      -type="oracle_database"
      -credentials="UserName:newuser;password:PWD_FILE;Role:SYSDBA"
      -input_file="PWD_FILE:at_pwd_file"
      -on_agent
```

The following example modifies the display name and properties for the `oracle_database` target with the name `database`. The `on_agent` flag ensures that the changes are propagated to the Management Agent collecting for this target.

```
emcli modify_target
      -name="database"
      -type="oracle_database"
      -display_name="New Name DB"
      -properties="SID=newsid|Port=15091|OracleHome=/oracle"
      -properties="MachineName=smpamp-sun1.us.oracle.com"
      -separator=properties="|"
      -subseparator=properties="="
      -on_agent
```

The following example modifies an `oracle_database` target type with the name `payroll_db`. In this example, the display name for this database (target name that is displayed in the Enterprise Manager UI) is being changed to `payroll`. The port number is being changed to `15067`, and the Oracle Home is being changed to `/oradb`. The administrator (`dbsnmp`), whose previous default role was `normal`, is being changed to `sysdba`. This example also illustrates the use of the `input_file` to camouflage the credentials. The password is actually in a file named `at_pwd_file`. The `-input_file` argument replaces `PWD_FILE` with the contents of `at_pwd_file` in the `-credentials` option.

```
emcli modify_target
      -name="payroll_db"
      -type="oracle_database"
      -credentials="UserName:Fred;password:PWD_FILE;Role:sysdba"
      -properties="Port:15067;OracleHome:/oradb"
      -input_file="PWD_FILE:at_pwd_file"
      -display_name=payroll
```

```
-on_agent
```

modify_user

Modifies an existing Enterprise Manager administrator.

Format

```
modify_user
      -name="name"
      [-password="password"]
      [-roles="role1;role2;..."]
      [-email="email1;email2;..."]
      [-privilege="name;[[target_name:target_type]|jobid]"]...

[ ] denotes that the parameter is optional
```

Options

- **name**

  Administrator name.

- **password**

  Replace administrator password with the specified password.

- **roles**

  Replace current roles with the specified list of Enterprise Manager roles to grant to this administrator. Currently, the built-in roles include PUBLIC.

- **email**

  Replace current email addresses for this administrator with the specified list. To delete all email addresses for this administrator, specify an empty string.

- **privilege**

  A privilege to grant to this administrator.

  You can specify this option more than once. The original administrator privileges will be revoked.

  The following system privileges do not require a target or a job ID:

  – `CREATE_ANY_ROLE`

  – `CREATE_ANY_PRIVILEGE`

  – `MANAGE_CREDENTIAL_GROUP`

  – `CREATE_TARGET`

  – `DELETE_ANY_TARGET`

  – `VIEW_ANY_TARGET`

  – `USE_ANY_BEACON`

  – `EM_MONITOR`

  – `SUPER_USER`

  The following target privileges require specifying `target_name:target_type`:

  – `VIEW_TARGET`

  – `OPERATOR_TARGET`

- MAINTAIN_TARGET
- CLONE_FROM_TARGET
- FULL_TARGET

The following group privileges require specifying `target_name:target_type`:

- CREATE_TARGET_IN_GROUP

The following job privileges require specifying `jobid`:

- VIEW_JOB
- FULL_JOB

Examples

The following example modifies the `new_admin` administrator. The user will have two privileges: to view the job with ID `923470234ABCDFE23018494753091111` and to view the target `host1.us.oracle.com:host`. The user will also be granted role `PUBLIC`. The user email addresses will be set to `first.last@oracle.com` and `joe.shmoe@shmoeshop.com`.

```
emcli modify_user
      -name="new_admin"
      -password="oracle"
      -email="first.last@oracle.com;joe.shmoe@shmoeshop.com"
      -roles="public"
      -privilege="view_job;923470234ABCDFE23018494753091111"
      -privilege="view_target;host1.us.oracle.com:host"
```

The following example deletes all the email addresses and privileges for administrator `new_admin`. Note that `-privilege=""` and `-privilege` are equivalent if specified at the command line in a UNIX shell.

```
emcli modify_user
      -name="new_admin"
      -email=""
      -privilege=""
```

provision

Provisions a hardware server, using configuration properties from the input file. The configuration properties required for a component can be viewed from the Grid Control console. After you make a provisioning request, you can view the status of the request from the Enterprise Manager Grid Control console by using the assignment name (specified by you or the automatically generated name returned to you).

Format

```
provision
      -image="path to the image"
      -network="network profile path"
      -bootserver="boot server name"
      -stageserver="stage server name"
      -stgcredentials="username"
      -schedule="type:immediate/onetime;timezone:zone;
startdt:startdate;starttm:time"
      -resettimeout="time"
      -target="hardware server label"
      -input_file="config_properties:file_path"
      -assignment= "assignment name"
      [-desc= "assignment description"]

[ ] denotes that the parameter is optional
```

Options

- **image**

  Path to the image-includes the image name. This is the image used for provisioning.

- **network**

  Path Name of network profile.

- **bootserver**

  Name of the boot server.

  Format: hostName:Directory Path

- **stageserver**

  Name of the stage server. hostName:Directory Path.

- **Stgcredentials**

  User name of the stage server.

- **schedule**

  Time when provisioning should be scheduled. This is a string argument that contains multiple name-value pairs separated by `;'. This is used to schedule the provisioning operation. "type" can be `immediate' or `onetime'. If "type" is not immediate, the other values are expected in the Time Zone: string, which is a timezone ID of the format:

  zone Sign TwoDigitHours:Minutes

  zone: Time zone ID (GMT, PDT, and so forth)

  Sign: one of "+ -"

TwoDigitHours: Digit Digit

Minutes: Digit Digit

Digit: One of 0 1 2 3 4 5 6 7 8 9

Startdt: Date string of the format: MM/DD/YY

Starttm: Time string of the format: HH:MM

- **resettimeout**

  Reset timeout for the hardware server in minutes.

- **target**

  Target hardware server is specified using the hardware label type.

- **input_file**

  File containing configuration properties.

- **assignment**

  Name of the assignment.

- **desc**

  Assignment description. The description is automatically generated if not specified.

Examples

The following example submits a job to provision `myimage` on a target with the label of `mylabel`. The job runs immediately with a reset timeout of 100 minutes. Image properties are picked from `properties.txt` that overrides the default image. `properties.stageserver` is used as the staging server, and `/private/share` as the staging storage with `joe` as the user name.

```
emcli provision
      -image="Images/myimage"
      -network="Networks/networkprofile"
      -bootserver="booservername.us.oracle.com"
      -stageserver="stageserver.us.oracle.com:/private/share"
      -stgcredentials="joe"
      -schedule="type:immediate"
      -resettimeout="100"
      -target="mylabel"
      -input_file="config_properties:properties.txt"
      -assignment="provision mylabel"
```

relocate_targets

There are two varieties of this Verb:

- The first variety creates a list of targets on the destination agent that already exists and is monitored by the source agent in Enterprise Manager. It moves all the collections and blackouts for these targets from the source agent to the destination agent, and makes the destination agent the monitoring agent for these targets in Enterprise Manager.

  ```
  emcli relocate_targets -src_agent=<source agent>
        -dest_agent=<destination agent>
        -input_file=dupTarget:<complete path to file>;
  ```

- The second variety makes the destination agent the monitoring agent for this target.

  ```
  emcli relocate_targets
        -src_agent=<source agent target name>
        -dest_agent=<destination agent target name>
        -target_name=<target name>
        -target_type=<target type>
        {-force=yes};
  ```

Format

```
relocate_targets
      -src_agent=<source agent target name>
      -dest_agent=<destination agent target name>
      {-target_name=<name of target to be relocated>
      -target_type=<type of target to be relocated> } |
      {-input_file=dupTarget:<complete path to file>}
      {-force=yes};
```

Options

- **src_agent**

  Agent currently monitoring the targets specified in the first argument.

- **dest_agent**

  Agent for which the Enterprise Manager user wants to monitor the targets specified in the first argument.

- **target_name**

  Name of the target that needs to be moved.

- **target_type**

  Type of target that needs to be moved.

- **input_file**

  Takes a file name that contains all the targets and its properties as seen in `targets.xml`. The contents of the file must be in same format as `targets.xml`.

- **force**

  This switch is optional. If the command is executed with the `-force=yes` switch, the composite target would be automatically relocated with its related targets. If the command is executed without this switch, an appropriate error message is displayed if it is a composite target.

remove_beacon

Removes a beacon from the monitoring set of beacons.

Format

```
remove_beacon
      -name=target name
      -type=target type
      -bcnName=beacon name
      [-forceRemove]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **bcnName**

  If specified, skips the sanity checks for availability definition.

Examples

The following example removes MyBeacon from MyTarget service target of type generic_service.

```
emcli remove_beacon -name='MyTarget' -type='generic_service'
      -bcnName='MyBeacon'
```

remove_service_system_assoc

Removes the system for a given Service.

Format

```
remove_service_system_assoc
      -name='name'
      -type='type'
```

Options

- **name**

  Service name.

- **type**

  Service type.

Examples

The following example removes the system for the generic service named 'my service'.

```
emcli remove_service_system_assoc
      -name='my service' -type='generic_service'
```

retry_job

Restarts a previously failed job execution.

Format

```
retry_job
      -exec_id="executionID"
      [-noheader]
      [-script | -format=
      [name:<pretty|script|csv>];
            [column_separator:"column_sep_string"];
            [row_separator:"row_sep_string"];
      ]

[ ] denotes that the parameter is optional
```

Options

- **exec_id**

  ID of the job execution to be retried. Use the get_jobs verb to obtain specific job execution Ids.

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to -format="name:script".

- **format**

  Format specification (default is -format="name:pretty").

  – format="name:pretty" prints the output table in a readable format not intended to be parsed by scripts.

  – format="name:script" sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – format="name:csv" sets the column separator to a comma and the row separator to a newline.

Output Columns:

Execution ID

Examples

The following example restarts the job execution with Id 12345678901234567890123456789012 and displays a new execution ID.

```
emcli retry_job -exec_id=12345678901234567890123456789012
```

revoke_privs

Revokes the privileges from an existing Enterprise Manager User or Enterprise Manager Role.

Format

```
emcli revoke_privs -name="username/rolename"
    [-privilege="name;[[target_name:target_type]|jobid]"]...
```

Options

- **name**

  User Name or Role Name from which privileges will be revoked.

- **privilege**

  Privilege, which will be revoked from Enterprise Manager User or Role. You can specify this option more than once.

  The following system privileges do not require a target or a job ID

  CREATE_ANY_ROLE

  CREATE_TARGET

  DELETE_ANY_TARGET

  VIEW_ANY_TARGET

  USE_ANY_BEACON

  EM_MONITOR

  The following target privileges require specifying target_name:target_type

  VIEW_TARGET

  OPERATOR_TARGET

  FULL_TARGET

  The following group privileges require specifying target_name:target_type

  CREATE_TARGET_IN_GROUP

  The following job privileges require specifying jobid

  VIEW_JOB

  FULL_JOB

Examples

```
1. emcli revoke_privs
       -name="user1"
       -privilege="FULL_JOB;923470234ABCDFE23018494753091111"
       -privilege="FULL_TARGET;host1.us.oracle.com:host"
   Revoke the privileges from Enterprise Manager user : user1
   Two privileges are revoked from user1 :
   1. Full control on the Jobs with ID 923470234ABCDFE23018494753091111
   2. Full control on the target host1.us.oracle.com:host

2. emcli revoke_privs
       -name="Role1"
```

```
          -privilege="FULL_TARGET;host1.us.oracle.com:host"
Revoke the target privileges from EM Role : Role1
```

revoke_roles

Revokes the roles to existing Enterprise Manager User or Enterprise Manager Role.

Format

```
emcli revoke_roles -name="username/rolename"
      [-roles="role1;role2;..."]

[ ] denotes that the parameter is optional
```

Options

- **name**

  User Name or Role Name from which roles will be revoked.

- **roles**

  Roles, which will be revoked from Enterprise Manager User or Role. You can specify this option more than once.

Examples

```
1. emcli revoke_roles
        -name="user1"
        -roles="SUPER_USER"

2. emcli revoke_roles
        -name="Role1"
        -roles="BLACKOUT_ADMIN;MAINTAIN_TARGET"
```

set_availability

Changes the availability definition of a given service.

Format

```
emcli set_availability
      -name=target name
      -type=target type
      -availType=availability type (can be 'test' or 'system')
      -availOp=availability operator (can be 'and' or 'or)
```

Options

- **-name=target name**

  Service target name.

- **-type=target type**

  Service target type.

- **-availType=availability type**

  Switches the availability to either test-based or system-based.

- **-availOp=availability operator**

  If `and`, it uses all key tests/components to decide availability.

  If `or`, it uses any key tests/components to decide availability.

Examples

```
emcli set_availability -name='MyTarget' type='generic_service'
                       -availType='test' -availOp='and'
Sets the availability of service MyTarget to be based on all key-tests.

emcli set_availability -name='MyTarget' type='generic_service'
                       -availType='test' -availOp='or'
Sets the availability of service MyTarget to be based on any key-test.
```

set_credential

Sets preferred credentials for given users.

Format

```
set_credential
      -target_type="ttype"
      [-target_name="tname"]
      -credential_set="cred_set"
      [-user="user"]
      -columns="col1:newval1;col2:newval2;..."
      [-input_file="tag1:file_path1;tag2:file_path2;..."]
      [-oracle_homes="home1;home2"]

[ ] denotes that the parameter is optional
```

Options

- **target_type**

  Type of target. The must be "host" in case the `-oracle_homes` parameter is specified.

- **target_name**

  Name of the target. Omit this argument to set enterprise preferred credentials. This must be the host name in case the `-oracle_homes` parameter is specified.

- **credential_set**

  Credential set affected.

- **user**

  Enterprise Manager user whose credentials are affected. If omitted, the current user's credentials are affected.

- **columns**

  Name and new value of the column(s) to set. Every column of the credential set must be specified. Alternatively, a tag from the `-input_file` argument can be used so that the credential values are not seen on the command line. You can specify this argument more than once.

- **input_file**

  Path of file that has the `-columns` argument(s). This option is used to hide passwords. Each path must be accompanied by a tag referenced in the `-columns` argument. You can specify this argument more than once.

- **oracle_homes**

  Name of oracle homes on the target host. Credentials will be added/updated for all specified homes.

Note: The list of columns and the credential sets they belong to is included in the metadata file for each target type. This and other credential information is in the `<CredentialInfo>` section of the metadata.

Examples

Example 1:

```
emcli set_credential
      -target_type=oracle_database
      -target_name=myDB
      -credential_set=DBCredsNormal
      -user=admin1
      -column="username:joe;password:newPass;role:newRole"
```

Example 2:

In Example 2, `FILE1` is a tag used to refer to the contents of `passwordFile`. Note that Example 2 has the same effect as Example 1.

```
emcli set_credential
      -target_type=oracle_database
      -target_name=myDB
      -credential_set=DBCredsNormal
      -user=admin1
      -column=FILE1
      -input_file=FILE1:passwordFile
```

Contents of the `passwordFile`:
`username:joe;password:newPass;role:newRole`

Example 3:

```
emcli set_credential
      -target_type=host
      -target_name=host.us.oracle.com
      -credential_set=OHCreds
      -user=admin1
      -column="OHUsername:joe;OHPassword:newPass"
      -oracle_homes="database1;mydb"
```

set_key_beacons_tests

Defines key beacons and tests of the service.

Format

```
set_key_beacons_tests
      -name=target name
      -type=target type
      [-beacons=beacon names]+
      [-tests='test1:type1;test2:type2;...']+
      [-removeKey]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **beacons**

  Names of beacons to set as key (or non-key).

- **tests**

  Names and types of tests to set as key (or non-key).

- **removeKey**

  If specified, the mode is (remove key); that is, the specified tests and beacons will be set as non-key.

  If not specified, the mode is (add key); that is, the specified tests and beacons will be set as key.

Examples

The following example sets `MyTest/HTTP`, `MyTest2/FTP` and `MyBeacon` as non-key elements of service `MyTarget/generic_service`.

```
emcli set_key_beacons_tests -name='MyTarget' -type='generic_service'
      -tests='MyTest:HTTP;MyTest2:FTP'
      -beacons='MyBeacon' -removeKey
```

The following example sets `MyBeacon` and `MyBeacon2` as key beacons of service `MyTarget/generic_service`.

```
emcli set_key_beacons_tests -name='MyTarget' -type='generic_service'
      -beacons='MyBeacon;MyBeacon2'
```

set_metric_promotion

Creates or edits a metric promotion based on a test or system.

Format

```
set_metric_promotion
      -name=Service target name
      -type=Service target type
      [-category = Usage/Performance]
      -basedOn = system/test
      -aggFunction = AVG|MAX|MIN|SUM|COPY
      [-promotedMetricName = Promoted Metric]
      [-promotedMetricColumn = Promoted Metric Column]
      -promotedMetricKey = Key Value of the promoted metric
      [-metricName = Dependent Metric Name]
      -column = Dependent Metric Column
      *[-depTargetType = Target type of dependent targets]
      *[-depTargets = 'target1;target2...']
      *[-depTargetKeyValues='target1:key11|key12|key13..;
target2:key21|key22|key23..']
      *[-depMetricKeyColumn= Dependent metric key column]
      **[-testname= Dependent Test Name]
      **[-testtype= Dependent Test Type]
      **[-metricLevel= TXN|STEP|STEPGROUP]
      **[-beacons='bcn1;bcn2..']
      **[-depTestComponent= Step or stepgroup name]
      [-threshold= 'Critical threshold value; Warning threshold value; Threshold
Operator (EQ|LE|LT|GT|GE)']
      -mode= CREATE|EDIT

[ ] denotes that the parameter is optional
```

`*` — Might be required if `basedOn` is set to `system`.

`**` — Might be required if `basedOn` is set to `test`.

Options

- **category**

  Defines whether the promoted metric is a usage or performance metric of a service. Category is used to determine the promoted metric name and metric column. If you do not specify this option, you must specify the `promotedMetricName` and `promotedMetricColumn` options.

- **basedOn**

  Determines whether the promotion is test-based or system-based.

- **aggFunction**

  Determines the aggregate function that will be used to compute the promoted metric `AVG/MAX/MIN/SUM` takes average, max, min, and sum of the dependent metrics, respectively. `COPY` only copies over a single dependent metric to the promoted metric.

- **promotedMetricName**

  Promoted metric name. This is optional if the category is specified.

- **promotedMetricColumn**

Promoted metric column. This is optional if the category is specified.

- **promotedMetricKey**

  Required argument that determines the key value of the promoted metric. It is equivalent to the displayed name of the promoted metric in the UI.

- **metricName**

  Required argument if the dependent metric column is collected by more than one metric.

- **column**

  Dependent metric column.

- **depTargetType**

  All dependent targets should be of this target type.

- **depTargets**

  Specifies the dependent targets. This argument is ignored if you specify `depTargetKeyValues`.

- **depTargetKeyValues**

  Specifies the key values associated with the dependent targets. Specify multiple key values for a single target by repeating the entry in the following format: `'tgt1:key1;tgt1:key2...'`

- **depMetricKeyColumn**

  Required if the dependent metric is a transpose metric. It is the key value that applies to all the dependent targets.

- **testname**

  Defines the name of the test to be used in promoting the metric.

- **testtype**

  Defines the type of the test to be used in promoting the metric.

- **metricLevel**

  Some metrics can be promoted on step-level. This option defines the level to be used during promotion.

- **beacons**

  List of beacons to be used for promoting the metric data.

- **depTestComponent**

  If `metricLevel` is not `TXN`, this option is required to specify which step or which step group is being promoted.

- **threshold**

  Defines a threshold on the promoted metric.-mode: Mode can be `CREATE` or EDIT.

Examples

The following example creates a promoted Performance metric with key value `mymetric1` on service `MyTarget` using `MyTest/HTTP`. The promoted metric takes the maximum of the `dns_time` metric column returned by the `MyBeacon` and `mybcn1` beacons. It also has a threshold with 'greater or equal to' operator (GE) with the critical value set to 200 and warning value set to 100.

```
emcli set_metric_promotion -name='MyTarget' -type='generic_service'
      -category=Performance -basedOn=test -aggFunction=MAX
      -testname='MyTest' -testtype=HTTP
      -beacons='MyBeacon, mybcn1'
      -promotedMetricKey=mymetric1 -column=dns_time -metricName=http_response
      -metricLevel=TXN -threshold='200;100;GE' -mode=CREATE
```

The following example creates a promoted Usage metric with key value `mymetric1` on service `MyTarget`. The dependent target is `'myhost.mydomain.com'` with type `host`. The promoted metric just copies the `cpuUtil` column of the `Load` metric.

```
emcli set_metric_promotion -name='MyTarget' -type='generic_service'
      -category=Usage -basedOn=system -aggFunction=COPY
      -promotedMetricKey=mymetric1 -column=cpuUtil -metricName=Load
      -depTargets='myhost.mydomain.com' -depTargetType=host
      -mode=CREATE
```

The following example creates a promoted `Usage` metric with the key value `AppServerComponentUsage` on service `MyTarget`. The dependent target is `'myapp_server'` with type `'oracle_ias'`. The promoted metric computes the average value of the `cpu.component` metric column for the specified key values.

```
emcli set_metric_promotion -name='MyTarget' -type='generic_service'
      -category=Usage -basedOn=system -aggFunction=AVG
      -promotedMetricKey=AppServerComponentUsage -depTargetType=oracle_ias
      -column=cpu.component
      -metricName=opmn_process_info
      -depTargetKeyValues='myapp_server:petstore;myapp_server:http_server'
      -mode=CREATE
```

set_properties

Sets the property for a test or beacons.

Format

```
set_properties
      -name=target name
      -type=target type
      -testname=test name
      -testtype=test type
      [-beacons=beacon names]
      [-properties='prop1:value1;prop2:value2;..']+

[ ] denotes that the parameter is optional
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **testname**

  Name of the test to set the property on.

- **testtype**

  Type of the test to set the property on.

- **beacons**

  Names of the beacons to set the property on.

- **properties**

  Names and values of the properties to be set (can be multiple).

Examples

The following example sets the property `timeout` to `30000` and granularity to `transaction` for the test `MyTest` defined on `MyTarget` for all beacons.

```
emcli set_property -name='MyTarget' -type='generic_service'
      -testname='MyTest' -testtype='HTTP'
      -propertyName='timeout:30000;granularity:transaction'
```

The following example sets the property value to `30000` of the test `MyTest` defined on `MyTarget` for only `MyBeacon` and `MyBeacon2`. This only works if the specified properties can be set on a per beacon level.

```
emcli set_property -name='MyTarget' -type='generic_service'
      -testname='MyTest' -testtype='HTTP'
      -bcnName='MyBeacon;MyBeacon2'
      -propertyName='timeout' -propertyValue='30000'
```

setup

Configures EM CLI to work with a specific management server.

Format

```
setup
  -url="http[s]://host:port/em/"
  -username=<EM Console Username>
  [-dir=<local emcli configuration directory>]
  [-trustall]
  [-novalidate]

[ ] denotes that the parameter is optional
```

Options

- **url="http[s]://host:port/em/"**

  URL of the Oracle management server (OMS). `host` specifies the host of the OMS. `port` specifies the listening port of the OMS. Both http and https protocols are supported.

- **username**

  Enterprise Manager user name to be used by all subsequent emcli commands when contacting the OMS.

- **dir**

  Directory where an EMCLI configuration directory will be created. This directory must be on a locally mounted file system. A warning and confirmation is issued for an HTTPS URL if the directory is not heuristically identified as such (unless you specify `trustall`). The directory can be relative to the working directory where setup is called, or it can be absolute. This option defaults to the user's home directory.

- **trustall**

  Automatically accept any server certificate from the OMS (lower security).

- **novalidate**

  Do not authenticate the Enterprise Manager user name against the OMS. Assume the given user name is valid.

Examples

```
emcli setup -url=http://myworkstation.us.oracle.com:7770/em -username=sysman
```

start_paf_daemon

Starts the Deployment Procedure Manager Daemon.

Format

```
emcli start_paf_daemon -interval={number in minutes}
```

Options

- **interval**

  Number in minutes that Deployment Procedure Manager Daemon should wait between each run.

status_paf_daemon

Gets the Deployment Procedure Manager Daemon status.

Format

```
emcli status_paf_daemon
```

Options

None.

stop_blackout

Stops a blackout.

You can stop a blackout before it has fully started, for example, when it has a "Scheduled" status. You can also stop a blackout while it is in effect.

Format

```
stop_blackout
      -name="name"
      [-createdby="blackout_creator" (default is current user)]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the blackout to stop.

- **createdby**

  Enterprise Manager user who created the blackout. The `SUPER_USER` privilege is required to stop a blackout created by another user.

Examples

The following example stops blackout `backup_db3` created by the current user.

```
emcli stop_blackout -name=backup_db3
```

The following example stops blackout `weekly_maint` created by user `joe`. The current user must either be user `joe` or a user with the `SUPER_USER` privilege.

```
emcli stop_blackout -name=weekly_maint -createdby=joe
```

stop_job

Stops a specified job. You can use the get_jobs Verb to obtain a list of job IDs and names.

Format

```
stop_job
      -job_id="jobID" | -name="jobName"
```

Options

- **job_id**

  Job ID to identify the job to stop.

- **name**

  Name of the job to stop. To uniquely identify the job, the current administrator is used.

Examples

The following example stops a job with the specified ID.

```
emcli stop_job -job_id=12345678901234567890123456789012
```

The following example stops a job named Backup_Wednesday, which is owned by the current Enterprise Manager administrator and scheduled to execute in the future.

```
emcli stop_job -name=Backup_Wednesday
```

stop_paf_daemon

Stops the Deployment Procedure Manager Daemon.

Format

```
emcli stop_paf_daemon
```

Options

None.

submit_job

Creates and submits a job.

Format

```
submit_job
      -job="name:type"
      -targets="name1:type1;name2:type2;..."
      -parameters="name1:value1;name2:value2;..."
      [-input_file="parameter_tag:file_path"]
      [-desc="job_description"]
      [-schedule=
            [frequency:<once|interval|weekly|monthly|yearly>];
            [start_time:<yy-MM-dd HH:mm>];
            [end_time:<yy-MM-dd HH:mm>];
            [repeat:<#m|#h|#d|#w|#M|#Y>];
            [months:<#,#,...>];
            [days:<#,#,...>];
            [tzoffset:#|[-][HH][:mm]]
            [tzinfo:<repository|target|specified>];
      ]
      [-noheader]
            [-script | -format=
                  [name:<pretty|script|csv>];
                  [column_separator:"column_sep_string"];
                  [row_separator:"row_sep_string"];
            ]
```

`[ ] denotes that the parameter is optional`

Constraints on schedule arguments:

`frequency:once`
optional => start_time, tzinfo, tzoffset

`frequency:interval`
requires => repeat

optional => start_time, end_time, tzinfo, tzoffset

`frequency:weekly`
requires => days

optional => repeat in #w, start_time, end_time, tzinfo, tzoffset

`frequency:monthly`
requires => days

optional => repeat in #M, start_time, end_time, tzinfo, tzoffset

`frequency:yearly`
requires => days, months

optional => repeat in #Y, start_time, end_time, tzinfo, tzoffset

Options

- **job**

  `name` represents the name for the submitted job.

type represents the type of the submitted job. The supported job types are OSCommand and SQLScript, which are already pre-defined in the Enterprise Manager job system. The specified job type determines which targets and which parameters can be specified for the -targets and -parameters arguments.

- **targets**

  A list of target-name, target-type pairs. The newly submitted job will apply to this list of Enterprise Manager targets. All targets must be of the same type. The target list must not contain more than one element if the element's target type is group. The OSCommand jobs are allowed to be submitted against targets of type host, oracle_database, and group (if it contains host targets). The SQLScript jobs are allowed to be submitted against targets of type oracle_database and group.

- **parameters**

  List of name-value pairs that represent the parameters required by the job type for this job. The OSCommand jobs support the parameters named command, args, os_script, username, password, and credential_set_name. command is the only required parameter.

  The SQLScript jobs support the parameters named sql_script, db_username, db_password, db_role, host_username, host_password, and credential_set_name. The required parameter is sql_script.

  The credential_set_name parameter refers to the set name of the preferred credentials stored in the Enterprise Manager repository. For each target type, several credential sets exist:

  – HostCredsNormal — Default unprivileged credential set for a host target

  – HostCredsPriv — Privileged credential set for a host target

  – DBHostCreds — Host credential set for an oracle_database target

  – DBCredsNormal — Default normal credential set for an oracle_database target

  – DBCredsSYSDBA: sysdba credential set for an oracle_database target

  You can only specify the credential_set_name parameter when the override credential parameters such as [db_|host_]username and [db_|host_]password are not present. If provided, the override credential parameters must be specified fully for each job type. For the OSCommand type, username and password must be specified together. For the SQLScript type, db_username, db_password, db_role, host_username, and host_password must be present.

- **input_file**

  Used in conjunction with the -parameters option, this option enables you to store specific job parameter values, such as passwords or SQL scripts, in a separate file. The -input_file option specifies a mapping between a tag and a local file path. The tag is specified in lieu of specific job parameter values of the -parameters option. The tag must not contain colons (:) or semi-colons (;).

- **desc**

  Job description.

- **schedule**

Job schedule. The `frequency` argument determines which other arguments are required or optional.

- **schedule=frequency**

  The type of job schedule (default is `once`).

- **schedule=start_time**

  Start date and time of the job. The default value is the current date/time. The format of the value is "yy-MM-dd HH:mm". For example: "2007-09-25 18:34".

- **schedule=end_time**

  Last date and time of the job. No job executions are scheduled after this date and time. When `frequency` is weekly, monthly, or yearly, only the date portion is used. When `frequency` is `interval` or `once`, the date and time are taken into account. The format of the value is "yy-MM-dd HH:mm". For example: "2007-09-25 18:34".

- **schedule=repeat**

  Time between successive start times when the job is scheduled. The letter following the number value represents the time units: "m" is minutes, "h" is hours, "d" is days, "w" is weeks.

- **schedule=months**

  List of integer month values in the range 1-12. Each value must have a corresponding "day" value, to fully specify (month,day) pairs that indicate the days of the year the job is scheduled.

- **schedule=days**

  When `frequency` is weekly, this is a list of integer day-of-week values in the range 1-7 (1 is Sunday). When `frequency` is monthly, this is a list of integer day-of-month values in the range 1-31 or -1 (last day of month). When `frequency` is yearly, this is a list of integer day-of-month values in the range 1-31 or -1 (last day of month); in this case, the month is taken as the corresponding `month` value for each (month,day) pair.

- **schedule=tzinfo**

  The type of timezone. The `tzinfo` argument is used in conjunction with `tzoffset`. Available timezone types are: `specified` (offset between GMT and the target timezone), `target` (timezone of the specified target), and `repository` (repository timezone — default setting when `tzinfo` is not specified). See `-schedule=tzoffset` for more information.

- **schedule=tzoffset**

  Value of the timezone. When you do not specify the `tzinfo` argument or it is `repository`, the `timezone` value is the repository timezone. In this case, you must not specify the `tzoffset` argument. Otherwise, the `tzoffset` argument is required. When you set `tzinfo` to `specified`, the `tzoffset` argument specifies the offset in hours and minutes between GMT and the timezone. When you set `tzinfo` is set to `target`, the `tzoffset` argument specifies an integer index (the first is 1) into the list of targets passed as arguments. For example, for a `tzoffset` setting of 1, the timezone of the first target specified in the `-add_targets` option is used.

  Note that the timezone is applied to the start time and the end time of the job schedule. The timezones associated with each target are not taken into account when scheduling the job (except that when you set `tzinfo` to `target`, the specified target's timezone is used for the job schedule).

- **noheader**

  Display tabular information without column headers.

- **script**

  This option is equivalent to `-format="name:script"`.

- **format**

  Format specification (default is `-format="name:pretty"`).

  – `format="name:pretty"` prints the output table in a readable format not intended to be parsed by scripts.

  – `format="name:script"` sets the default column separator to a tab and the default row separator to a newline. The column and row separator strings can be specified to change these defaults.

  – `format="name:csv"` sets the column separator to a comma and the row separator to a newline.

Output Columns

Job ID, Execution ID

Examples

The following example submits a job that runs `ls -l` against target `"hostname.oracle.com:host"`. The job runs under OS username `joe` with a password of `greetings`.

```
emcli submit_job
      -job="job_host_0:OSCommand"
      -parameters="command:ls;args:-l;username:joe;password:greetings"
      -targets="hostname.us.oracle.com:host"
```

The following example submits a job that runs the shell (`/bin/sh)` script specified by the parameter `large_os_script` against targets `hostname1.oracle.com:host` and `hostname2.oracle.com:host`. The targets' preferred credentials are used to run this job. Here, `large_os_script` can be up to 4 GB.

```
emcli submit_job
      -job="job_host_1:OSCommand"
      -parameters='command:/bin/sh;args:-x;large_os_script:LARGE_SCRIPT_FILE'
      -input_file="LARGE_SCRIPT_FILE:very_large_os_script.sql"
      -targets="hostname1.oracle.com:host;hostname2.oracle.com:host"
```

The following example submits a job that runs the SQL script specified in the file `./very_large_script.sql` against the target `database:oracle_database`. The target's preferred credentials are used to run this job. Here, `large_sql_script` can be up to 4 GB.

```
emcli submit_job
      -job="job_db_1:SQLScript"
      -parameters="large_sql_script:LARGE_SQL_FILE"
      -targets="database:oracle_database"
      -input_file="LARGE_SQL_FILE:very_large_script.sql"
```

submit_procedure

Submits a Deployment Procedure.

Format

```
emcli submit_procedure
     -procedure="guid of the procedure"
     -input_file="data:{file_path}"
     [-instance_name="name for the procedure instance"]
     [-schedule=start_time:yyyy/MM/dd HH:mm;tz:{java timezone ID}];]

[ ] denotes that the parameter is optional
```

Options

- **procedure**

  GUID of the procedure to execute.

- **input_file=data:file_path**

  Input data for the Deployment Procedure. The `file_path` should point to a file containing the data XML file.

- **instance_name**

  Optional name of the procedure instance.

- **schedule**

  Optional schedule for the Deployment Procedure. If not specified, the procedure is executed immediately.

  `start_time` — When the procedure should start

  `tz` — Optional time zone ID

Output Columns

Instance GUID

Examples

```
emcli submit_procedure -input_file=data:data.xml
     -procedure=16B15CB29C3F9E6CE040578C96093F61 -schedule="start_time:2006/6/21
      21:23;tz:America/New_York"
```

subscribeto_rule

Subscribes the user to a rule with email notification.

It is not an error to specify email addresses that are already in the `assignto` user's preferences.

A message appears if the outgoing mail server (SMTP) has not been set up. When you specify the option `-fail_if_no_mail_server`, this condition is an error and prevents the subscribe from occurring; otherwise, this condition is a warning that does not affect the success of this command.

Format

```
subscribeto_rule
      -name="rule_name"
      -owner="rule_owner"
      [-assignto="em_username" (default is current user)]
      [-email="email_address";...]
      [-fail_if_no_mail_server]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the notification rule.

- **owner**

  Owner of the notification rule.

- **assignto**

  User to subscribe to the notification rule. If the `assignto` user is not the current user, or if the owner of the rule is not the current user, the super-user privilege is needed.

- **email**

  List of email addresses to associate with the rule to which the `assignto` user is being subscribed. These addresses are first added to the preferences of the `assignto` user (duplicates are ignored) before being assigned to the notification rule. The email addresses are added only if the current user has the privilege to subscribe the `assignto` user to the rule.

- **fail_if_no_mail_server**

  A message appears if the outgoing mail server (SMTP) has not been set up. When you specify the option `-fail_if_no_mail_server` is specified, this condition is an error and prevents the subscribe from occurring; otherwise, this condition is a warning that does not affect the success of this command.

Examples

The following example subscribes the current user to the rule `"Agent Upload Problems"` using the current user's email addresses for notification. The current user must have the `SUPER_USER` (or have `sysman`) privilege for this to succeed, since `sysman` owns the rule. Also, the current user must already have at least one email address in his/her preferences for this command to succeed.

```
emcli subscribeto_rule -name="Agent Upload Problems" -owner=sysman
```

The following example first adds the two specified email addresses to the preferences for user `joe`. Then user `joe` is subscribed to the rule `"Agent Upload Problems"` using joe's email addresses for notification. The current user must have the `SUPER_USER` privilege (or be `joe`) for this command to succeed.

```
emcli subscribeto_rule -name="Agent Upload Problems" -owner=sysma
      -assignto=joe -email="joe@work.com;joe@home.com"
```

sync

Synchronizes the EMCLI client with an OMS. After synchronization, all Verbs and associated command line help available to this OMS become available at the EMCLI client.

Synchronization occurs automatically during a call to setup.

Format

```
sync
```

Options

None.

Examples

```
emcli sync
```

sync_beacon

Synchronizes a beacon that is monitoring the target (reloads all collections to beacon).

Format

```
sync_beacon
      -name=target name
      -type=target type
      -bcnName=beacon name
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **bcnName**

  Beacon name to synchronize.

Examples

The following example synchronizes `MyBeacon`, which is monitoring `MyTarget` target of type `generic_service`.

```
emcli sync_beacon -name='MyTarget' -type='generic_service'
      -bcnName='MyBeacon'
```

update_password

Updates passwords (or other credentials) for a given target.

Format

```
update_password
      -target_type="ttype"
      -target_name="tname"
      -credential_type="cred_type"
      -key_column="column_name:column_value"
      -non_key_column="col:oldvalue:newvalue;..."
      [-input_file="tag1:file_path1;tag2:file_path2;..."]

[ ] denotes that the parameter is optional
```

Options

- **target_type**

  Type of target.

- **target_name**

  Name of the target.

- **credential_type**

  Credential type to use. The type must be a base type, not a derived type. A derived type contains the XML tag `<CredentialTypeRef>` within its definition.

- **key_column**

  Name and value of the key column for the credential type. Usually, the key column represents the user name.

- **non_key_column**

  Name, old value, and new value of the non-key column(s) to modify. Usually, this is the name of the password column. Alternatively, you can use a tag from the `-input_file` argument so that the credential values are not seen on the command line. You can specify this argument more than once.

- **input_file**

  Path of the file that has `-non_key_column` argument(s). This option is used to hide passwords. You must accompany each path by a tag referenced in the `-non_key_column` argument. You can specify this argument more than once.

Note: The list of columns and the credential types they belong to is included in the metadata file for each target type. This and other credential information is in the `<CredentialInfo>` section of the metadata.

Host Example

For credentials associated with host targets, use the following arguments for the command.

```
target_type=host
credential_type = HostCreds
key_column=HostUserName:<OSUserName>
non_key_column=HostPassword:<oldPassword>:<newPassword>
```

The following example changes the password associated with the OS user `sysUser` from `sysUserOldPassword` to `sysUserNewPasword` in all features of EM that use this OS username. This includes preferred credentials, corrective actions, jobs, and OS user-defined metrics.

```
update_password   -target_type=host  -target_name=MyHost  -credential_
type=HostCreds  -key_column=HostUserName:sysUser
-non_key_column=HostPassword:sysUserOldPassword:sysUserNewPassword
```

Oracle Database Examples

For credentials associated with database targets, use the following arguments for the command.

```
target_type=oracle_database
credential_type = DBCreds
key_column=DBUserName:<DBUser>
non_key_column=DBPassword:<oldPassword>:<newPassword>  OR
non_key_column=DBPassword:<oldPassword>:<newPassword>:<DBRole>
```

The following example changes the password associated with the database user `scott` from `tiger` to `tiger2` for all features of Enterprise Manager that use this database username. This includes preferred credentials, corrective actions, jobs, SQL user-defined metrics, and the monitoring configuration for this database target in Enterprise Manager.

```
update_password   -target_type=oracle_database  -target_name=ORCL  -credential_
type=DBCreds  -key_column=DBUserName:scott
-non_key_column=DBPassword:tiger:tiger2
```

The following example changes the password associated with the database user `sys` from `sysPassword` to `sysNewPassword` for all features of Enterprise Manager that use this database username. This includes preferred credentials, corrective actions, jobs, SQL user-defined metrics, and the monitoring configuration for this database target in Enterprise Manager.

```
update_password   -target_type=oracle_database  -target_name=ORCL  -credential_
type=DBCreds  -key_column=DBUserName:sys
-non_key_column=DBPassword:sysPassword:sysNewPassword:DBAROLE
```

Oracle Listener Example

For credentials associated with Listener targets, use the following arguments for the command.

```
target_type=oracle_listener
credential_type = LsnrCreds
key_column (not applicable)
non_key_column=Password:<oldPassword>:<newPassword>
```

The following example changes the password associated with the Listener from `oldListenerPassword` to `newListenerPassword` for all features of Enterprise Manager that use this password. This includes preferred credentials, corrective actions, jobs, and the monitoring configuration for this Listener target in Enterprise `Manager`.

```
update_password   -target_type=oracle_listener  -target_name=MyListener
-credential_type=LsnrCreds
-non_key_column=Password:oldListenerPassword:newListenerPassword
```

3 Error Code Reference

This chapter covers errors and associated codes returned by EM CLI. You can use EM CLI return codes to manage the control flow in a workflow/scripting environment. EM CLI return codes for Verb errors are positive integers. A Verb returns either 0 (successful execution) or an error number.

The following sections provide reference tables for these types of errors:

- EM CLI infrastructure
- OMS connection
- File-fed option
- Built-in verb

3.1 EM CLI Infrastructure Errors

Any execution of the EM CLI client could result in the following errors.

Table 3–1 Infrastructure Errors

Error Code	Description
242	A Verb has encountered a problem with a dependency specific to the implementation of the Verb (INSIDE of its abstraction barrier) unrelated to the Verb's semantics.
248	Configuration files are corrupt or inaccessible.
253	The command name is not recognized.
254	Internal system error.

3.2 OMS Connection Errors

Verbs that execute at the OMS return these error codes as indicated in the listing for each applicable Verb.

Table 3–2 OMS Connection Errors

Error Code	Description
243	License has not been accepted by the current user.
249	Cannot connect to the OMS.
250	Wrong credentials for log in to the OMS.

3.3 File-fed Option Errors

Verbs that allow for file-fed options (rather than options where the values are explicitly defined on the command line) can return the following error codes.

Table 3–3 File-Fed Option Errors

Error Code	Description
244	Cannot find an option value file.
245	Cannot read in an option value file.
246	An option value file is too big.

3.4 Built-in Verb Errors

The following error codes are returned by each Verb (not including EM CLI infrastructure errors that apply to all Verbs).

Table 3–4 Built-In Verb Errors

Verb	Error Code
add_beacon	0—Beacon added successfully.
	129— Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	170—Service does not exist.
	173—Beacon does not exist.
	201—Beacon is already in the monitoring beacons list.
	230—Insufficient privileges.
	255—Back-end error. Verb failed.
add_group_to_mpa	2—I/O error occurred while writing to the MPA file.
	3—The specified MP already exists in the MPA.
	4—The group name is empty or not specified.
	223—The supplied options are syntactically incorrect.
add_mp_to_mpa	1—File does not exist, is unreadable, or an I/O error occurred.
	2—I/O error occurred while writing to the MPA file.
	3—The specified MP already exists in the MPA.
	4—The target-type definition file cannot be parsed.
	5—The MPA filename is not between 1 and 255 characters.
	6 —A file of a particular file type is required for another file.
	223—The supplied options are syntactically incorrect.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
add_target	1—The supplied target type does not exist. Unable to retrieve target metadata from the specified host's Management Agent. 2—Host does not exist. 3—Agent does not exist. 4—Group does not exist. 5—No monitoring credentials set found for target in the repository. 6—Target instance already exists in the repository. 7—The supplied target properties are incomplete. 8—One or more of the supplied target properties are invalid. 15—Target deletion in progress. 20—Unable to connect to the specified host's Agent. 21—Unable to save the target instance to the specified host's Agent. 22—Cannot add more than one Agent target for a single Agent URL. 23—Unable to add an instance of an Agent target without a URL. 219—Insufficient privileges to add the target to the group. 223—Unable to parse command line correctly. Invalid argument value. File-Fed Option Errors—The errors associated with file-fed options. OMS Connection Errors—The errors associated with connecting to the executing OMS.
apply_privilege_delegation_setting	0—Setting successfully applied. 2—Setting does not exist. 3—All or some of the targets are invalid. 129—Syntax error. The displayed message indicates which argument is syntactically incorrect.
apply_template_tests	1—Error processing input XML file. 4—Insufficient privileges for apply template. 6—Target does not exist. 7—Incompatible template and target types during apply. 8—Test(s) specified for overwriteExisting do not exist in the template. 9—Key test(s) specified as disabled for apply. 10—Stepgroup contains a step that does not exist in the file. 11—Some text property in file does not conform to valid syntax. 12—Some text property contains variable but variable value is missing. 13—Some transaction property/threshold/collection setting does not conform to required restrictions. 50—Generic error.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
argfile	Possible return error codes consist of the following list plus all of the errors returned by the Verb specified in the command line file for execution. 244—The file does not exist. 245—There is a problem reading in the file or it does not exist. 246—The file ends inside a quoted token. 247—The argfile options are specified incorrectly.
assign_test_to_target	0—Test assigned to target type successfully. 129—Syntax Error. The displayed message indicates which argument is syntactically incorrect. 190—Test or target type invalid. 230—Insufficient privileges. 255—Back-end error. Verb failed.
change_service_system_assoc	0—Service system changed successfully. 129—Syntax Error. The displayed message indicates which argument is syntactically incorrect. 170—Service does not exist. 171—System <system> does not exist. 172—Key component does not exist. 230—Insufficient privileges. 255—Back-end error. Verb failed.
clone_as_home	1—The source_params parameter is invalid or in wrong format. Example: Source Home location, hostname are missing. 2—Destination properties file format is invalid. 3—Source Home/software library data invalid. No Source Home/software library fetched from the repository matches data specified by user. 4—Product type does not match the specified cloning verb. Example: Attempted to clone a database but specified an Application Server as a source. 5—Invalid input parameters specified. This is a generic error message for all cases not covered by the previous error messages. In some cases, the parameter itself may be in a valid format, but may point to a home that is not readable or corrupt. 6—Error validating Destination home. 7—Error validating/collecting information from Source Home. This error is typically returned during Application Server cloning when the Application Server properties file cannot be read from the Source Home. 8—Other internal error occurred: Exceptions within cloning APIs, or validation, database access APIs.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
clone_crs_home	1—The source_params parameter is invalid or in wrong format. Example: Source Home location, host name are missing. 2—Destination properties file format is invalid. 3—Source Home/software library data is invalid. No Source Home/software library fetched from the repository matches data specified by user. 4—Product type does not match the cloning verb used. Example: Attempted to clone a database, but supplied an Application Server as a source. 5—Invalid input parameters specified. Generic error message for all cases not covered by previous error messages. In some situations, the parameter itself may be in a valid format, but may point to a home that is not readable or corrupt. 6—Error validating Destination home. 7—Error validating/collecting information from Source Home. This error is typically returned during Application Server cloning when the Application Server properties file cannot be read from the Source Home. 8—Other internal error occurred. Exceptions raised within cloning APIs, or validation database access APIs.
clone_database_home	1—The source_params parameter is invalid or in wrong format. Example: Source Home location, host name are missing. 2—Destination properties file format is invalid. 3—Source Home/software library data invalid- no Source Home /software library fetched from the repository matches data specified by user. 4—Product type does not match the cloning verb used. Example: You attempted to clone a database but specified an Application Server as a source. 5—Invalid input parameters specified: generic error message for all cases not covered above. In some cases, the parameter itself may be in a valid format, but may point to a home that is not readable or corrupt. 6—Error validating Destination home. 7—Error validating/collecting information from Source Home: This error is typically returned during Application Server cloning when the Application Server properties file cannot be read from the Source Home. 8—Other internal error occurred: Exceptions within cloning APIs, or validation, database access APIs.
create_aggregate_service	1—Target does not exist. 2—Target exists.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
create_blackout	1—Blackout X already exists. 2—Only Super Administrators are allowed to add a new reason (use get_ blackout_reasons). 3—Agent targets cannot be directly blacked out. 217—The blackout end_time cannot be in the past. The dates specified will never cause this blackout to take effect. The difference between the end_time and the start_time must be equal to the duration. The difference between the repeat interval and the duration must be at least X minutes. The duration must be -1 (for indefinite blackouts) or positive. The duration must be at least X minutes. 219—Current user does not have OPERATOR privilege over all blackout targets. 220—Target X does not exist. 223—Unable to parse command line correctly. OMS Connection Errors—The errors associated with connecting to the executing OMS.
create_group	1—Group X already exists. 2—Cannot add target X to typed group of base type Y. 218—Group X is currently in the process of being deleted. 219—Current user does not have privilege X over all member targets. 220—Member target X does not exist. 223—Unable to parse command line correctly. Invalid argument value. Group type is invalid. OMS Connection Errors—The errors associated with connecting to the executing OMS.
create_privilege_ delegation_setting	0—Setting successfully created. 129—Syntax error. The displayed message indicates which argument is syntactically incorrect.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
create_red_group	0—Redundancy Group "<red_group_name>" created successfully.
	1—Redundancy Group "<red_group_name>" of target type <red_group_type> already exists.
	2—Cannot add target "<member_target_type>" to typed group of base type "<red_group_type>".
	3—Time Zone Region <timezone_region> does not exist.
	4—Redundancy Group Type "<red_group_type>" is invalid.
	218—Redundancy Group "<red_group_name>:<red_group_type>" is currently in the process of being deleted.
	220—Target "<member_target_name>:<member_target_type>" does not exist.
	223—Redundancy Group name "<red_group_name>" is not valid. It may contain only alphanumeric characters, multi-byte characters, a space, "-", "_", ".", ":", and have a maximum length of 256 characters.
	223—User name "<owner>" is not valid. It must begin with an alphabetic character, contain only alphanumeric characters, underscores (\"_\"), or periods (\".\"), and have a maximum length of 256 characters.
	223—Invalid value for parameter "add_targets": "<add_targets>". Reason: "<add_targets>" is not a name-value pair.
	223—Member Targets not of same type.
	223—"<generic_redundancy_group>" does not support member of type "<member_target_type>" .
create_role	1—Role by same name already exists.
	2—User with same name as role already exists.
	4—Privilege is invalid or nonexistent.
	5—Target specified in one of the privileges is invalid.
	6—The Super Administrator privilege cannot be granted to a role.
	7—Role does not exist.
	8—Group specified in one of the privileges is invalid.
	9—Job in privilege is invalid or nonexistent.
	10—Creating a role that you are assigning to the new role.
	11—The specified user does not exist.
	219—User is unauthorized to perform this action.
	223—Unable to parse command line correctly.
	Invalid argument value.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
create_service	0—Web application created successfully.
	129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	130—Missing key components.
	151—Test validation failed.
	171—System <system> does not exist.
	172—Key component does not exist.
	173—Beacon does not exist.
	181—No key tests defined.
	182—No key beacons defined.
	200—Service <target_name> already exists.
	230—Insufficient privileges.
	255—Back-end error. Verb failed.
create_system	0—System "<system_name:system_type>" created successfully.
	110—System "<system_name:system_type>" already exists.
	120—Member target "<member_target_name>:<member_target_type>" does not exist.
	122—Type "<system_type>" is not a valid System type.
	123—Time Zone Region "<timezone_region>" does not exist.
	130—Type meta version "<type_meta_ver>" is invalid.
	223—System name "<system_name>" is not valid. It must begin with an alphabetic char, contain only alphanumeric chars or any of "- _.:", and have a maximum length of 256 chars.
	223—Type meta version "<type_meta_ver>" is invalid. It must contain only numeric and "." characters, and have a maximum length of 8 chars.
	223—Timezone_region cannot be null or blank.
	223—Invalid value for parameter "add_members": "<add_members>". Reason: "<add_members>" is not a name-value pair.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
create_user	1—Target specified in one of the privileges is invalid.
	2—Group specified in one of the privileges is invalid.
	3—Job specified in one of the privileges is invalid.
	4—One of the specified privileges is invalid.
	5—Such user already exists.
	6—One or more roles to be granted to the new user does not exist.
	7—A role with the same name as the new user already exists.
	218—A delete is pending against this user until all blackouts and jobs submitted by this user are stopped.
	219—User has insufficient privileges to perform this operation.
	223—Unable to parse command line correctly:
	Invalid argument value.
	User name is somehow invalid.
	Supplied password does not have the proper format. Example: Password left empty.
	File-Fed Option Errors—The errors associated with file-fed options.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
delete_blackout	1—Blackout X created by user Y does not exist.
	2—Cannot delete a blackout that has not ended or was not stopped.
	219—You (X) do not have the SUPER_USER privilege needed to stop, delete, or modify blackout Y created by user Z.
	Only the blackout owner can stop, delete, or modify the blackout.
	Current user does not have OPERATOR privilege over all blackout targets.
	223—Unable to parse command line correctly.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
delete_group	1—Group X does not exist.
	218—Group X is currently in the process of being deleted.
	219—Current user does not have sufficient privileges to perform this action.
	223—Unable to parse command line correctly.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
delete_job	1—Specified job is invalid or non-existent.
	219—User has insufficient privileges to perform this operation.
	218—Some executions are not stopped when delete happens.
	223—Unable to parse command line correctly.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
delete_metric_promotion	0—SUCCESS 223—SYNTAX_ERRNUM: Input is malformed. 255—VERB_FAILED_ERRNUM: Back-end validation fails.
delete_privilege_delegation_settings	0—Setting successfully deleted. 2—All or some of the names are invalid. 129—Syntax error. The displayed message indicates which argument is syntactically incorrect.
delete_role	1—Role does not exist. 219—User is unauthorized to perform this action. 223—Unable to parse command line correctly. OMS Connection Errors—The errors associated with connecting to the executing OMS.
delete_system	0—System "<system_name:system_type>" deleted successfully. 121—System "<system_name:system_type>" does not exist. 122—Type "<system_type>" is not a valid System type. 219—Current user does not have sufficient privileges to perform this action. 223—System name "<system_name>" is not valid. It must begin with an alphabetic character, contain only alphanumeric characters or any of "- _.:", and have a maximum length of 256 chars.
delete_target	15—Target deletion in progress. 219—Insufficient privileges to delete specified target. 220—Target does not exist. 223—Unable to parse command line correctly. OMS Connection Errors—The errors associated with connecting to the executing OMS.
delete_test	0—Test deleted successfully. 129—Syntax Error. The displayed message indicates which argument is syntactically incorrect. 170—Service does not exist. 174—Test does not exist. 230—Insufficient privileges. 255—Back-end error. Verb failed.
delete_user	1—Cannot delete the repository owner. 2—Specified user does not exist. 3—Cannot delete the current user. 218—A delete is pending against this user until all blackouts and jobs submitted by this user are stopped. 219—User has insufficient privileges to perform this operation. 223—Unable to parse command line correctly. OMS Connection Errors—The errors associated with connecting to the executing OMS.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
disable_test	0—Test disabled successfully.
	129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	170—Service does not exist.
	174—Test does not exist
	203—Test already disabled.
	230—Insufficient privileges.
	255—Back-end error. Verb failed.
enable_test	0—Test enabled successfully.
	129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	170—Service does not exist.
	174—Test does not exist
	202—Test already enabled.
	230—Insufficient privileges.
	255—Back-end error. Verb failed.
execute_hostcmd	0—Command execution succeeded for all targets.
	2—Command execution failed for one or more targets. Detailed errors will be displayed for each failed target.
	3—Invalid or unknown targets in the targets list.
	4—Preferred credentials are missing for one or more targets.
	5—Invalid credential set name.
	223—Unable to parse the command line properly.
execute_sql	0—Command execution succeeded for all targets.
	2—Command execution failed for one or more targets. Detailed errors will be displayed for each failed target.
	3—Invalid or unknown targets in the targets list.
	4—Preferred credentials are missing for one or more targets.
	5—Invalid credential set name.
	223—Unable to parse the command line properly.
export_template	223—Unable to parse command line correctly, or an exception was thrown during SQL handling.
	245—There is a problem writing to the file.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
extend_as_home	1—The source_params argument is invalid or in the wrong format. Example: Source Home location or host name are missing. 2—Destination properties file format is invalid. 3—Source Home/software library data invalid. No Source Home/software library fetched from the repository matches data specified by user. 4—Product type does not match the cloning verb used. Example: Attempted to clone a database but specified an Application Server as a source. 5—Invalid input parameters specified. Generic error message for all cases not covered by previous error messages. In some cases, the parameter itself may be in a valid format, but may point to a home that is not readable or corrupt. 6—Error validating destination home. 7 —Error validating/collecting information from source home. Typically returned during Application Server cloning when the Application Server properties file cannot be read from the Source Home. 8—Other internal error occurred: Exceptions within cloning APIs, or validation, database access APIs.
extend_crs_home	1—The source_params parameter is invalid or in the wrong format. Example: Source Home location or host name are missing. 2—Destination properties file format is invalid. 3—Source Home/software library data invalid. No Source Home/software library fetched from the repository matches data specified by user. 4—Product type not matching with the cloning verb used. Example: Attempted to clone a database, but specified an Application Server as a source. 5—Invalid input parameters specified. Generic error message for all cases not covered by previous error messages. In some cases, the parameter itself may be in a valid format, but may point to a home that is not readable or is corrupt. 6—Error validating destination home. 7—Error validating/collecting information from Source Home: Typically returned during Application Server cloning when the Application Server properties file cannot be read from the Source Home. 8—Other internal error occurred: Exceptions within cloning APIs, or validation, database access APIs.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
extend_rac_home	1—The source_params parameter is invalid or in wrong format. Example: Source Home location, host name are missing. 2—Destination properties file format is invalid. 3—Source Home/software lib data invalid- no Source Home /software library fetched from the repository matches data specified by user. 4—Product type does not match the cloning verb used. Example: tried to clone database, but gave app server as source. 5—Invalid input parameters specified: generic error message for all cases not covered above. In some cases the parameter itself may be in a valid format, but may point to a home which is not readable or corrupt. 6—Error validating destination home. 7—Error validating/collecting information from Source Home: Typically returned during Application Server cloning when the Application Server properties file cannot be read from the Source Home. 8—Other internal error occurred: Exceptions within cloning APIs, or validation, database access APIs.
extract_template_tests	2—Error serializing XML output. 3—Insufficient privileges for extract template. 5—Template does not exist in repository. 50—Generic error.
get_aggregate_service_info	1—Target does not exist. 2—Target exists.
get_aggregate_service_members	1—Target does not exist. 2—Target exists.
get_blackout_details	1—Blackout X created by user Y does not exist. 223—Unable to parse command line correctly. OMS Connection Errors—The errors associated with connecting to the executing OMS.
get_blackout_reasons	OMS Connection Errors—The errors associated with connecting to the executing OMS.
get_blackout_targets	1—Host X does not exist. 223—Unable to parse command line correctly. 220—Target X does not exist.
get_blackouts	1—Host X does not exist. 220—Target X does not exist. 223—Unable to parse command line correctly. OMS Connection Errors—The errors associated with connecting to the executing OMS.
get_group_members	1—Group X does not exist. 223—Unable to parse command line correctly. OMS Connection Errors—The errors associated with connecting to the executing OMS.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
get_groups	Other than the confirmation message, the get_groups verb only generates syntax errors. The SQL invoked by get_groups does not throw any exception.
	0—All groups (TargetName, TargetType) in the repository are displayed.
	223—Syntax Error: Argument -script cannot be specified with a value.
	223—Syntax Error: -format argument "name" value must match one of these strings: "script|pretty|csv".
	223—Syntax Error: Invalid value for parameter "format": "name:<format_name>;column_separator=<column_separator_char>". Reason: "column_separator=column_separator_char" is not a name-value pair.
	223—Syntax Error: -format argument contains an unrecognized key name <key_name>
get_jobs	223—Unable to parse command line correctly.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
get_system_members	121—System "<system_name:system_type>" does not exist.
get_targets	223—Unable to parse command line correctly.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
help	1—There is no help available.
	223—Unable to parse the command line correctly.
import_template	21—Occurs if one of the templates has an OMS version specified in it that does not match the version of the OMS you are importing it into, and there are no other errors.
	22—Occurs if one of the template files cannot be parsed, and there are no other errors.
	99—More than one of the templates to be imported had errors during processing.
	223—Unable to parse command line correctly, or an exception was thrown during SQL handling.
	245—There is a problem reading in the file, or it does not exist.
modify_aggregate_service	1—Target does not exist.
	2—Target exists.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
modify_group	1—Group X does not exist.
	2—Cannot add target X to typed group of base type Y.
	3—Group X contains itself as a sub-group at some level.
	219—Current user does not have sufficient privileges to perform this action:
	Current user does not have privilege X over all member targets. Current user does not have sufficient privileges on target X to add it to the group.
	220—Target X does not exist.
	223—Unable to parse command line correctly. Group type is invalid.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
modify_red_group	0—Redundancy Group ""<red_group_name>" modified successfully.
	1—Redundancy Group ""<red_group_name>:<red_group_type>" does not exist.
	2—Cannot add target "<member_target_type>" to typed group of base type "<red_group_type>".
	4—Redundancy Group Type "<red_group_type>" is invalid.
	218—Redundancy Group "<red_group_name>:<red_group_type>" is currently in the process of being deleted.
	220—Target "<member_target_name>:<member_target_type>" does not exist.
	223—Redundancy Group name "<red_group_name>" is not valid. It may contain only alphanumeric characters, multi-byte characters, a space, "-", "_", ".", ":", and have a maximum length of 256 characters.
	223—User name "<owner>" is not valid. It must begin with an alphabetic character, contain only alphanumeric characters, underscores (\"_\"), or periods (\".\"), and have a maximum length of 256 characters.
	223—Invalid value for parameter "add_targets": "<add_targets>". Reason: "<add_targets>" is not a name-value pair.
	223—Member Targets not of same type.
	223—"Generic redundancy group" does not support member of type "<member_target_type>" .

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
modify_role	4—Privilege is invalid or nonexistent.
	5—Target specified in one of the privileges is invalid.
	6—The Super Administrator privilege cannot be granted to a role.
	7—Role does not exist.
	8—Group specified in one of the privileges is invalid.
	9—Job in privilege is invalid or nonexistent.
	10—Cannot have a circular chain of role grants.
	11—The specified user does not exist.
	219—User is unauthorized to perform this action.
	223—Unable to parse command line correctly. Invalid argument value.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
modify_system	0—System "<system_name:system_type>" modified successfully.
	101—System <system_name:system_type> contains itself as a sub-system at some level.
	120—Member target "<member_target_name>:<member_target_type>" does not exist.
	121—System "<system_name:system_type>" does not exist.
	122—Type "<system_type>" is not a valid System type.
	219—Current user does not have sufficient privileges on target <member_target_name> to add it to the system.
	219—Current user does not have sufficient privileges to perform this action.
	223—Invalid value for parameter "add_members": "<add_members>". Reason: "<add_members>" is not a name-value pair.
modify_target	8—One or more of the supplied target properties are invalid.
	15—Target deletion in progress.
	219—Insufficient privileges to modify target.
	220—Target does not exist.
	223—Unable to parse command line correctly.
	File-Fed Option Errors—The errors associated with file-fed options.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
modify_user	1—Target specified in one of the privileges is invalid. 2—Group specified in one of the privileges is invalid. 3—Job specified in one of the privileges is invalid. 4—One of the specified privileges is invalid. 5—Specified user does not exist. 6—One or more roles to be granted to the new user does not exist. 218—A delete is pending against this user until all blackouts and jobs submitted by this user are stopped. 219—User has insufficient privileges to perform this operation. 223—Unable to parse command line correctly: Invalid argument value or user name is somehow invalid. File-Fed Option Errors—The errors associated with file-fed options. OMS Connection Errors—The errors associated with connecting to the executing OMS.
provision	1—An Internal error occurred. Could not get an Instance of the Assignment Manager. Exception occurred when getting URN from path. 2—Could not provision. Exception occurred either in getting editable ProvisioningAssigment object, or during call to Initiate Provisioning. 3—Could not get one or more URNs. Returned if any of imageUrn, bootServerUrn, stageServerUrn, networkProfileUrn, targetUrn retrieved is null. 4—Could not create assignment state. Failed to create an AssignmentState object. 5—Could not set assignment properties. Failed to set the assignment properties in the assignment state object. Since this verb uses the FileArgRemoteVerb, the following errors are also possible: ■ This Verb posts Verb.SYNTAX_ERRNUM if a specified option/file mapping on the command line is not properly formatted. ■ This Verb posts Verb.LOGIN_SYSTEM_ERRNUM if it cannot log in to the OMS. ■ This Verb posts Verb.OMS_CONNECTION_SYSTEM_ERRNUM if it cannot connect to the OMS. ■ This Verb posts Verb.CONFIGURATION_SYSTEM_ERRNUM if the configuration files are corrupt or inaccessible. ■ This Verb posts Verb.MISSING_FILE_SYSTEM_ERRNUM if it cannot find an option value file. ■ This Verb posts Verb.FILE_READ_SYSTEM_ERRNUM if it cannot read in an option value file. ■ This Verb posts Verb.FILE_SYNTAX_SYSTEM_ERRNUM.

Table 3–4 (Cont.) Built-In Verb Errors

<table>
<tr><th>Verb</th><th>Error Code</th></tr>
<tr><td>relocate_targets</td><td>
0—Moved all targets from Source Agent to Destination Agent.
1—Target relocation has failed. The following errors are possible:

SQL exception when relocating targets : <Database-specific error message>.
Communication exception when relocating targets: < communication exception message >.
Verb usage error:
<pre>emcli relocate_targets
   -src_agent=<source agent target name>
   -dest_agent=<dest agent target name>
    {-target_name=<name of the target to be
relocated>
   - target_type=<type of the target to be
relocated>}
| {-input_file=dupTargets:<complete path to file>}
  {-force=yes}; "</pre>
Errors relocating targets from Source Agent to Destination Agent:
< error message >
< error message >
Exception in parsing targets from the command line argument <message>.

</td></tr>
<tr><td>remove_beacon</td><td>
0—Beacon removed successfully.
129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
170—Service does not exist.
173—Beacon does not exist.
225—Beacon not in monitoring beacons list.
230—Insufficient privileges.
255—Back-end error. Verb failed.
</td></tr>
<tr><td>remove_service_system_ assoc</td><td>
0—System removed from service successfully.
129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
170—Service does not exist.
180—System does not exist.
230—Insufficient privileges.
255—Back-end error. Verb failed.
</td></tr>
<tr><td>retry_job</td><td>
1—Cannot restart job of a non-restartable type.
2—Specified job execution does not exist or has not failed.
3—The specified job execution has already been restarted and failed on restart.
219—User has insufficient privileges to perform this operation.
223—Unable to parse command line correctly.
OMS Connection Errors—The errors associated with connecting to the executing OMS.
</td></tr>
</table>

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
set_availability	0—Availability set successfully.
	129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	170—Service does not exist.
	180—No system defined.
	181—No key tests defined.
	182—No key beacons defined.
	230—Insufficient privileges.
	231—Availability not changed.
	255—Back-end error. Verb failed.
set_credential	1—Target type does not exist.
	2—Target (of given target type) does not exist.
	3—Credential set does not exist.
	4—Insufficient privileges.
	5—Credential column does not exist.
	6—Credential column number mismatch.
set_key_beacons_tests	0—Key beacons and tests set successfully.
	129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	135—Must specify at least one key beacon and test.
	170—Service does not exist.
	173—Beacon does not exist.
	175—Beacon not in list of monitoring beacons.
	230—Insufficient privileges.
	255—Back-end error. Verb failed.
set_metric_promotion	0—SUCCESS
	223—SYNTAX_ERRNUM: Input is malformed.
	255—VERB_FAILED_ERRNUM: Back-end validation fails.
set_properties	0—Properties set successfully.
	129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	132—Invalid property.
	133—Invalid property value.
	170—Service does not exist.
	173—Beacon does not exist.
	175—Beacon not in list of monitoring beacons.
	230—Insufficient privileges.
	255—Back-end error. Verb failed.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
setup	1—The Verb cannot establish a configuration area, or a corrupt area already exists.
	2—A connection with the OMS cannot be established.
	3—The login with the provided credentials fails at the OMS.
	4—The supplied "url" option is malformed or is not http/https.
	5—The configuration directory is not local as determined by the user in non-trustall HTTPS mode.
	6—The Verb cannot collect the user password safely.
	7—License is not been accepted by the user.
	223—Unable to parse command line correctly.
stop_blackout	1—Blackout X created by user Y does not exist.
	2—The blackout has already ended or stopped.
	3—Agent-side blackouts cannot be edited or stopped.
	218—The start of the blackout is currently being processed.
	The blackout is already pending stop.
	The last set of edits to the blackout have not yet been committed.
	219—You (X) do not have the Super Administrator privilege needed to stop, delete, or modify blackout Y created by user Z.
	Only the blackout owner can stop, delete, or modify the blackout.
	Current user does not have OPERATOR privilege over all blackout targets.
	223—Unable to parse command line correctly.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
stop_job	1—Specified job is invalid or non-existent.
	219—User has insufficient privileges to perform this operation.
	223—Unable to parse command line correctly.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
submit_job	1—Supplied job type is invalid or non-existent.
	2—Job with the same name already exists.
	3—One or more specified targets are invalid.
	4—Missing job parameter.
	5—Invalid job parameters, possibly including the security parameters such as "pwd".
	217—Specified job schedule is invalid.
	219—User has insufficient privileges to perform this operation.
	223—Unable to parse command line correctly.
	Invalid argument value.
	File-Fed Option Errors—The errors associated with file-fed options.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
subscribeto_rule	1—Rule with name X and owner Y does not exist.
	2—EM user X does not exist.
	3—EM user X has no email addresses set up (see console tab Preferences->General).
	4—Outgoing Mail (SMTP) Server not set up (see console tab Setup->Notification Methods).
	219—You (X) do not have the SUPER_USER or MANAGE_ ANY_ USER privilege needed to add email addresses for user Y.
	You (X) do not have the SUPER_USER or MANAGE_ANY_ USER privilege needed to subscribe Y to the rule owned by Z.
	223—Unable to parse command line correctly.
	Invalid argument value.
	OMS Connection Errors—The errors associated with connecting to the executing OMS.
sync	1—The Verb cannot establish a configuration area or a corrupt area already exists.
	2—A connection with the OMS cannot be established.
	3—The login with the provided credentials fails at the OMS.
	4—The license has not been accepted by the current user.
	223—Unable to parse the command line correctly.
sync_beacon	0—Beacon synced successfully.
	129—Syntax Error. The displayed message indicates which argument is syntactically incorrect.
	170—Service does not exist.
	173—Beacon does not exist.
	175—Beacon not in list of monitoring beacons.
	230—Insufficient privileges.
	255—Back-end error. Verb failed.
update_password	4—Target (of given target type) does not exist.
	5—Credential type does not exist for given target.
	6—Key value (that is, user name) does not exist.
	7—Non-operator cannot change credentials.
	8—Wrong value for old password.
	9—Old and new passwords match.
	10—No such non_key_column name.

Index