Oracle® Enterprise Manager

Command Line Interface

11*g* Release 1 (11.1)

E16185-08

May 2013

ORACLE®

Oracle Enterprise Manager Command Line Interface, 11*g* Release 1 (11.1)

E16185-08

Primary Author: Michael Zampiceni

Contents

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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

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 Concepts and Installation

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

1.1 Overview

The Enterprise Manager Command Line Interface (EM CLI) enables you to access Enterprise Manager Grid Control functionality from text-based consoles (shells and command windows) for a variety of operating systems. You can 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, such as monitoring and 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. You can also use EM CLI commands interactively from an operating system console.

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

Examples of EM CLI usage are as follows:

- Integrate Enterprise Manager with third-party or custom software through scripting. Actions (such as adding/deleting targets, submitting/deleting jobs, creating/deleting users) 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 a.m.

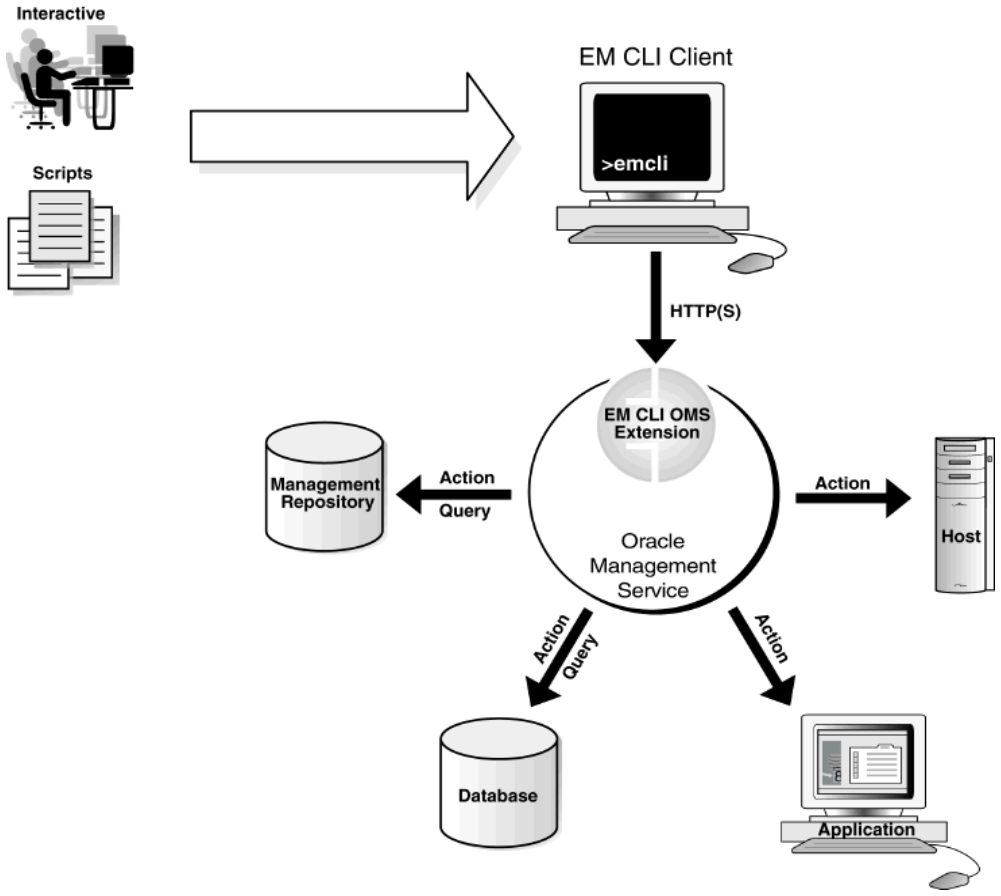
1.2 How the EM CLI Works

The EM CLI Client is a Java application that accepts a command as input. The EM CLI Client 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 Oracle Management Services (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 locally 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 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.
- To avoid an uncommon occurrence in which multiple emcli sessions are created on the OMS, Oracle recommends that you enter the `login` command before running a script containing EMCLI commands.

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 system within your managed network. The EM CLI Client is a command-line program (Java-based) that sends EM CLI verbs to a specific OMS. In some respects, the EM CLI Client functions as a command-line

equivalent of an Enterprise Manager Grid Control console. The EM CLI OMS Extension is automatically installed with the OMS and serves as the communication conduit between the EM CLI Client and the OMS.

1.4.1 Requirements

Before installing EM CLI, you will need the following items:

- Enterprise Manager 11g Grid Control framework
- Java version 1.6 or greater
- Workstation running Solaris, Linux, HPUX, Tru64, AIX, or Windows with NTFS (client installation)

1.4.2 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.

1.4.2.1 Installing the EM CLI Client

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

   You can download the EM CLI client kit from any 11.1 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 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 1.4.1 or greater. For example:

   ```
   setenv JAVA_HOME /usr/local/packages/j2sdk1.4.1_02

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

3. Enter the following command to ensure that you have the correct Java in your PATH:

   ```
   which java
   ```

   This should show the Java in `$JAVA_HOME/bin`.

1. Install the EM CLI Client. You can install the client portion of EM CLI in any directory either on the same system as the OMS or on any system in your network (download the `emclikit.jar` file to that system).

   Go to the directory where you have installed `emclikit.jar`:

   ```
   cd $HOME/<your emcli installation directory>
   ```

2. Enter 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 setting up the client.

1.4.2.2 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 installation directory 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.
- **login** — Log in and establish a session with the OMS.
- **logout** — Log out of EM CLI client from Enterprise Manager.
- **setup** — Configure EM CLI to function with a specific OMS.
- **sync** — Synchronize the EM CLI Client with an OMS.
- **version** — List EM CLI verb versions or the EM CLI client version.
- **add_mp_to_mpa** — Create (or add to) the Management Plug-in Archive. The Management Plug-in Archive is available for adding new target types to Enterprise Manager.
- **add_group_to_mpa** — Add a Management Plug-in group to a Management Plug-in Archive.

1.4.2.3 Running Setup

You must run `setup` to connect the EM CLI Client to the OMS running the EM CLI Management Services. 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.

1. Understand the syntax of the `setup` verb and its options by entering the following command or referring to the `setup` verb in the verb reference chapter of this guide:

   ```
   ./emcli help setup
   ```

2. Enter the `setup` verb with at least the minimum required parameters as shown in the following example:

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

   As you observed from step 1, the `setup` verb has several options, including the following important options:

   - `ssousername` and `ssopassword`
   - `noautologin`
   - `custom_attrib_file`

3. Enter your user password when prompted after the EM CLI client connects with the EM CLI Management Services.

After running the `setup` verb, the message "Emcli Setup Successful" appears, and you are ready to begin using EM CLI.

Tip: For complete information on the Setup verb and its options, including `ssousername`, `ssopassword`, `noautologin`, and `custom_attrib_file` referenced in step 2, see the setup verb on page 2-247.

To configure the EM CLI Client to function with multiple OMSes by implementing multiple setups, see the Examples section for the Setup verb.

1.4.2.4 Using 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 are necessarily present. Logs are created as needed and are append-based — they are preserved between invocations of EM CLI. You can safely delete log files any time without affecting EM CLI operation. The logs contain stack traces, which may not be useful for the casual user, but may benefit you 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 1–1 No configuration directory is specified with the setup verb (Default location)

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

If you do not specify a configuration directory when you run the `setup` verb (`-dir` option is omitted), EM CLI assumes the `.emcli` configuration directory is located within the your 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 1–2 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).

1.4.2.5 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 1–3 Setting EMCLI_OPTS in a Microsoft Windows Environment

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

Example 1–4 Setting EMCLI_OPTS in a UNIX Environment (TCSH)

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

1.4.3 EM CLI Command-line Help

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, enter the following command for an overview of all available verbs ...

```
./emcli help
```

... or enter the following command for a detailed verb description, verb arguments and options, and usage examples.

```
./emcli help <verb>
```

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 1–5 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 11g Release 11.1

Copyright (c) 1996, 2010 Oracle Corporation. All rights reserved.

CONFIG DIRECTORY: /home/emcli_install_dir/.emcli

OMS : http://my_system.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 "Using EM CLI Log Files" on page 1-6 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. See the `login` and `logout` verbs for information on credentials.
- Make credentials persistent on the host system 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.

1.5.3 Secure Mode for the EM CLI Setup

The EM CLI client installs certain configuration files and a client-side implementation of verbs on the client system. The EM CLI client configuration files contain information such as OMS URL, Enterprise Manager user name, Enterprise Manager password, and SSO user name and password (if EM is SSO-enabled). The default mode stores these credentials, which is inherently insecure because of backward compatibility reasons.

To eliminate this security risk, a secure mode EM CLI setup does not store any EM or SSO passwords on the client disk. For a secure setup, you need to specify the `noautologin` option for the Setup verb. You provide the credentials once during setup, after which a session is established between the client and OMS. All subsequent verbs use this session. Inactivity or an explicit logout (using the Logout verb) terminates this session, and a re-setup or an explicit login (using the Login verb) is required before invoking any new verb.

- For information on the noautologin option, see the setup verb on page 2-247.
- For information on logging in, see the login verb on page 2-173.
- For information on logging out, see the logout verb on page 2-175.

1.6 Script Availability for Output Data Verbs

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 1–6 Complex Separator

Separator Specification: `xxx\u0009xxx\u0009`

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

2

Verb Reference

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

2.1 Verb Categories

For your convenience, this section provides another method other than alphabetization for finding verbs. All of the verbs for this release are listed in their respective categories.

Agent Administration Verbs

get_agent_properties
get_agent_property
secure_agent
set_agent_property

Agent Patch Verbs

submit_agent_patch

Agent Recovery Verbs

resyncAgent

Audit Settings Verbs

disable_audit
enable_audit
show_audit_settings
show_operations_list
update_audit_settings

Basic Operational Verbs

Note: These verbs are the only ones available immediately after installation.

argfile
help
login
logout
setup
sync
version

Blackout Verbs

create_blackout
delete_blackout
get_blackout_details
get_blackout_reasons
get_blackout_targets
get_blackouts
stop_blackout

Cloning Verbs

clone_as_home
clone_crs_home
clone_database_home
extend_as_home
extend_crs_home
extend_rac_home

Credential Verbs

clear_credential
set_credential
show_credential_set_info
show_credential_type_info
update_host_password
update_password
update_target_password

Credential Verbs - Oracle Database

update_db_password

Deployment Procedure Verbs

confirm_instance
delete_instance
get_instance_data_xml
get_instance_status
get_instances
get_procedure_types
get_procedure_xml
get_procedures
get_retry_arguments
ignore_instance
reschedule_instance
resume_instance
retry_instance
set_instance_jobgrants
start_paf_daemon
status_paf_daemon
stop_instance
stop_paf_daemon
submit_procedure
suspend_instance
update_and_retry_step

Execute Command Verbs

execute_hostcmd
execute_sql

Group Verbs

create_group
delete_group
get_group_members
get_groups
modify_group

Job Verbs

delete_job
get_job_execution_detail
get_jobs
retry_job
stop_job
submit_job

Licensing Verbs

grant_license_no_validation
grant_license_with_validation
revoke_license_no_validation
revoke_license_with_validation

Management Plug-in Verbs

add_group_to_mpa
add_mp_to_mpa

Management Services and Repository Verbs

loader_perf

Masking Verbs

export_masking_definition
generate_masking_script
import_masking_definition
list_masking_definitions
reassoc_masking_definition
save_masking_script
submit_masking_job

Metric Collection and Alerts Verbs

clear_stateless_alerts
collect_metric
get_metrics_for_stateless_alerts
get_on_demand_metrics
get_unsync_alerts
modify_collection_schedule

Monitoring Templates Verbs

apply_template
export_template

import_template

Notification Verbs

subscribeto_rule

Privilege Delegation Settings Verbs

apply_privilege_delegation_setting
clear_privilege_delegation_setting
create_privilege_delegation_setting
delete_privilege_delegation_settings
list_privilege_delegation_settings
list_target_privilege_delegation_settings

Provisioning Verbs

provision

Redundancy Group Verbs

create_red_group
create_redundancy_group
modify_red_group
modify_redundancy_group
view_redundancy_group

Report Import/Export Verbs

export_report
get_reports
import_report

SecureComm Verbs

get_ca_info

Services Verbs

add_beacon
apply_template_tests
assign_test_to_target
change_service_system_assoc
create_aggregate_service
create_service
delete_metric_promotion
delete_test
delete_test_threshold
disable_test
enable_test
extract_template_tests
get_aggregate_service_info
get_aggregate_service_members
get_test_thresholds
modify_aggregate_service
remove_beacon
remove_service_system_assoc
run_avail_diag
run_promoted_metric_diag
set_availability

set_key_beacons_tests
set_metric_promotion
set_properties
set_test_threshold
sync_beacon

System Verbs

create_system
delete_system
get_system_members
modify_system

Target Data Verbs

add_target
add_target_property
delete_target
get_target_properties
get_targets
modify_target
relocate_targets
remove_target_property
set_target_property_value
set_standby_agent

User Administration Verbs

create_role
create_user
delete_role
delete_user
grant_privs
grant_roles
modify_role
modify_user
revoke_privs
revoke_roles

Virtualization Verbs

Note: Use of any of the following verbs requires that you install a one-off patch. Refer to the documentation in this chapter for any of the verbs in this list for more information.

delete_guest_vm
get_guest_vm_status
get_virtual_server_status
list_guest_vm
list_ovm_virtual_server_pool
list_virtual_server
list_virtual_server_pool
migrate_vsp_ovm_to_em
pause_guest_vm
reboot_guest_vm
reboot_virtual_server
resume_guest_vm
start_guest_vm

start_vt_daemon
status_vt_daemon
stop_guest_vm
stop_virtual_server
stop_vt_daemon
suspend_guest_vm
unpause_guest_vm

WebLogic Server Verbs

discover_wls

2.2 Alphabetical 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.

- view_redundancy_group on page 2-297

add_beacon

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

Format

```
emcli add_beacon
      -name=target name
      -type=target type
      -bcnName=beacon name
      [-dontSetKey]
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **bcnName**

  Beacon name to add.

- **dontSetKey**

  Indicates the added beacon is not automatically a key beacon. Only use this option if you do not want the beacon to participate in the availability calculation of the service and tests.

Examples

The following example adds MyBeacon to the 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

```
emcli 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**

  Name of the group.

- **member**

  An MP to be added to the group. The MP is specified by its target type as found in the target type definition file and version. Using "newest" as the version specifies no version for this group member MP. When operating on the group, the newest version of the MP available is used. The order of MPs define the order these MPs become deployed when the stack is deployed. The reverse order is used for undeployment.

- **desc**

  Description of the group.

Examples

Example 1

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"
```

Example 2

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:newest"
```

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

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

[ ] denotes that the parameter is optional
```

Options

- **mpa**

  Name of the MPA where the resulting MP is placed. The MPA file name can 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), indicates 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 — Script on the Agent executed by a job type.
- JOB_DEFINITION — XML file that defines a job type.
- HOMEPAGE_DEFINITION — XML file that defines charts to show on the home page.

You must specify JOB_SCRIPT and JOB_DEFINITION 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. Since this description is optional, 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 this 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.

> **Note:** You cannot use this verb for composite targets. The verb does not support adding an association between a parent target such as IAS and a child target such as OC4J.

Format

```
emcli add_target
      -name="name"
      -type="type"
      -host="hostname"
      [-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"]
      [-groups="groupname1:grouptype1;groupname2:grouptype2;..."]
      [-timezone_region="gmt offset"]
      [-monitor_mode="monitor mode"]
      [-instances="rac database instance target name1:target type1;..."]

[ ] 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 system 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 this file. Metadata files are located in `$AGENT_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**

  Specifies a string delimiter to use between the 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. 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 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**

  Target name 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.

- **instances**

  Name-value pair list of RAC database instances that the RAC database target has.

Examples

Example 1

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.example.com"
```

```
      -credentials="UserName:dbsnmp;password:dbsnmp;Role:Normal"
      -properties="SID:semcli;Port:15091;OracleHome:/oracle;
       MachineName:smpamp-example.com"
      -groups="Group1:database_group;Group2:group"
```

Example 2

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.example.com"
      -credentials="UserName:dbsnmp;password:PWD_FILE;Role:Normal"
      -properties="SID:semcli;Port:15091;OracleHome:/oracle;
       MachineName:example.com"
      -input_file="PWD_FILE:/emcli_dir/pwdfiles/at_pwd_file"
```

Example 3

The following example illustrates how to add a RAC database with given installed RAC database instances and clusterware. The example adds a `rac_database` target with the name cluster_database and the cluster name **newdb_cluster**. A RAC instance is picked up among instances on the given host. This verb should be called after database instances and clusterwares have been installed.

```
emcli add_target
      -name="cluster_database"
      -type="rac_database"
      -host="myhost.example.com"
      -monitor_mode="1"
      -properties="ServiceName:service.example.com;ClusterName:
       newdb_cluster"
      -instances="database_inst1:oracle_database;database_inst2:
       oracle_database"
```

Example 4

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.example.com"
      -properties="LsnrName:LISTENER;ListenerOraDir:/oracle/lsnr;
       Port:15091;OracleHome:/oracle;Machine:smpamp-sun1.us
```

add_target_property

Adds a new target property for a given target type. All targets of this target type will have this new target property.

Format

```
emcli add_target_property
         -target_type="target_type"
         -property="prop_name"
```

Options

- **target_type**

  Target type for which this property needs to be added. To add this property to all existing target types, you can specify a "*" wildcard character.

- **property**

  Name of the property to be created for this target type. Property names are case-sensitive. The property name cannot be the same as the following Oracle-provided target property names (in English):

  Comment, Deployment Type, Line of Business, Location, Contact

Examples

Example 1

The following example adds the owner name property for all targets of type oracle_database.

```
emcli add_target_property -target_type="oracle_database" -property="Owner Name"
```

Example 2

The following example adds the Owner property for all target types.

```
emcli add_target_property -target_type="*" -property="Owner"
```

apply_privilege_delegation_setting

Activates Sudo or PowerBroker settings for specified targets.

Format

```
emcli apply_privilege_delegation_setting
         -setting_name="setting"
         -target_type="host/composite"
         [-target_names="name1;name2;..."]
         [-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

Example 1

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;....."
```

Example 2

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
```

Example 3

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"]
        [-replace_metrics="0" or "1"]
        [-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`

  A 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 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'.

- **replace_metrics**

  0 indicates that the thresholds of the metrics not included in the template but available in the target will not be changed. This is the default option. 1 indicates that the thresholds of the metrics present in the target, but not in the template, will be set to NULL. That is, such metrics in the target will not be monitored and therefore, no alert will be raised for them.

- **input_file**

  You can use this parameter to specify the location of a file, which contains the credentials to be used for the User Defined Metrics (UDMs) if the template contains any UDMs. file_name actually refers to the name of the file along with the

path of the location, which contains the credentials applicable for the UDMs. For example:

```
emcli apply_template -name="template1"  -targets="mydb1:oracle_database"
  -input_file= "FILE1:/usr/template/apply_udm_credentials.txt"
```

This example applies a monitoring template named "template1" to target mydb1 of type oracle_database, and the credentials needed for the UDMs are accessed from the file "/usr/template/apply_udm_credentials.txt".

The contents of the file apply_udm_credentials.txt should be in one of the following formats:

– All UDMs use the same credentials for all targets. For example:

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

– Each UDM uses its own credentials for all targets. For example:

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

– Each UDM uses different credentials for different targets. For example:

```
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 to specify the "credListType" in every input text file that you specify.

Examples

Example 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 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"
```

Example 2

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"
```

Example 3

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 the 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

```
emcli 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=template**

  Name of the input file containing the test definitions.

- **input_file=variables**

  Name of the input file containing the variable definitions. If this attribute is not specified, the variables are extracted 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>, and so forth exist on the target, they are 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 the file `my_template.xml` into the Generic Service target `my_target`, using the key `my_password` to decrypt the file contents. If tests with names `my_website` or `my_script` exist on the target, they are 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 one or more EM CLI verbs, where both verbs and the associated arguments are contained in an ASCII file. argfile enables 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.

Multiple emcli verb invocations are permitted in this file. You should separate each verb invocation with a new line.

Format

```
emcli argfile /path/to/<input_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

```
emcli 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

```
emcli 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 the service resides.

- **systemtype**

  System type.

- **keycomponents**

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

Example

The following example changes the 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'
```

clear_credential

Clears preferred or monitoring credentials for given users.

Format

```
emcli clear_credential
      -target_type="<ttype>"
      [-target_name="<tname>"]
      -credential_set="<cred_set>"
      [-user="<user>"]
      [-oracle_homes="<home1;home2"]

[ ] denotes that the parameter is optional
```

Options

- **target_type**

  Type of target, which must be "host" if you specify the oracle_homes parameter.

- **target_name**

  Name of the target. Omit this option to clear enterprise-preferred credentials. The target name must be the host name if you specify the oracle_homes parameter.

- **credential_set**

  Credential set affected.

- **user**

  Enterprise Manager user whose credentials are affected. If omitted, the current user's credentials are affected. This value is ignored for monitoring credentials.

- **oracle_homes**

  Name of Oracle homes on the target host. Credentials are cleared for all specified homes.

Examples

```
emcli clear_credential
       -target_type=oracle_database
       -target_name=myDB
       -credential_set=DBCredsNormal
       -user=admin1


emcli clear_credential
       -target_type=oracle_database
       -credential_set=DBCredsNormal
       -user=admin1
```

clear_privilege_delegation_setting

Clears the privilege delegation setting from a given host or hosts.

Format

```
emcli clear_privilege_delegation_setting
        -host_names="name1;name2;..."
        [-input_file="FILE:file_path"]
        [-force="yes/no"]

[ ] denotes that the parameter is optional
```

Options

- **host_names**

  Names of the hosts.

- input_file

  Path of the file that has the list of hosts. The file should have one host name per line.

- force

  If set to yes, invalid and unreachable targets are ignored and the setting is removed from all valid and up targets. If set to no, invalid and down targets raise an error. The default is no.

Examples

Example 1

```
emcli clear_privilege_delegation_setting
      -host_names="host1;host2;....."
```

Example 2

```
emcli clear_privilege_delegation_setting
      -host_names="host1;host2;....."
      -force=yes
```

Example 3

```
emcli clear_privilege_delegation_setting
      -input_file="FILE:/home/user/file.txt"
      -force=yes
```

clear_stateless_alerts

Clears the stateless alerts associated with the specified target. Only the user can clear these stateless alerts; the Enterprise Manager Agent does not automatically clear these alerts. To find the metric internal name associated with a stateless alert, use the get_metrics_for_stateless_alerts verb.

Format

```
emcli clear_stateless_alerts
        -older_than=number_in_days
        -target_type=target_type
        -target_name=target_name
        [-include_members]
        [-metric_internal_name=target_type_metric:metric_name:metric_column]
        [-unacknowledged_only]
        [-ignore_notifications]
        [-preview]

[ ] denotes that the parameter is optional
```

Options

- **older_than**

  Specify the age of the alert in days. (Specify 0 for currently open stateless alerts.)

- **target_type**

  Internal target type identifier, such as host, oracle_database, and emrep.

- **target_name**

  Name of the target.

- **include_members**

  Applicable for composite targets to examine alerts belonging to members as well.

- **metric_internal_name**

  Metric to be cleaned up. Use the get_metrics_for_stateless_alerts verb to see a complete list of supported metrics for a given target type.

- **unacknowledged_only**

  Only clear alerts if they are not acknowledged.

- **ignore_notifications**

  Use this option if you do not want to send notifications for the cleared alerts. This may reduce the notification sub-system load.

- **preview**

  Shows the number of alerts to be cleared on the target(s).

Examples

The following example clears alerts generated from the database alert log over a week old. In this example, no notifications are sent when the alerts are cleared.

```
emcli clear_stateless_alerts -older_than=7 -target_type=oracle_database -tar
get_name=database -metric_internal_name=oracle_database:alertLog:genericErrStack
-ignore_notifications
```

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
       -isIas1013="true/false"
       [-oldIASAdminPassword=oldpass]
       [-newIASAdminPassword=newpass]
       [-oldoc4jpassword=oldpass]
       [-oc4jpassword=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

- **input_file="dest_properties:file_path"**

  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.

- **isIas1013**

  Specifies whether this is a 10.2.3 Ias home.

- **oldoc4jpassword**

  Old OC4j password. (Required for 10.1.3 Ias homes.)

- **oc4jpassword**

  New OC4J password. (Required for 10.1.3Ias homes.)

- **oldIASAdminPassword**

  Old Application Server administrator password. (Not required for 10.1.3 Ias homes.)

- **newIASAdminPassword**

  New Application Server administrator password. (Not required for 10.1.3 Ias homes.)

- **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.

Examples

```
emcli clone_as_home
   -input_file="dest_properties:/home/destinations.txt"
   -list_exclude_files="centralagents.lst"
   -isSwLib="false"
   -tryftp_copy="false"
   -jobname="clone as home"
   -iasInstance="asinstancename"
   -isIas1013="false"
   -oldIASAdminPassword="oldpassword"
   -newIASAdminPassword="newpassword"
   -prescripts="/home/abc/myscripts"
   -run_prescripts_as_root="true"
   -rootscripts="%oracle_home%/root.sh"
   -source_params="TargetName:host.domain.com;HomeLoc=/home/oracle/appserver1;
    HomeName=oracleAppServer1;ScratchLoc=/tmp"
```

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
       [-isWindows="false/true"]
       [-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

- **input_file="dest_properties:file_path"**

  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.

- **isWindows**

  Specify whether the cloning source is on a Windows Platform. This option only applies for creating CRS cloning from a Gold Image source. The default value is false.

- **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

- **input_file=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. 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.

- **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. 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, 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 `%%`.

collect_metric

Performs an immediate collection and threshold evaluation of a set of metrics associated with the specified internal metric name. You typically use this command when you believe you have resolved an open metric alert or error and would like to clear the event by immediately collecting and reevaluating the metric. This command applies to most metrics except server-generated database metrics.

Use the get_on_demand_metrics verb to see a complete list of supported metrics for a given target.

Format

```
emcli collect_metric
       -target_name=name
       -target_type=type
       -metric_name=metric_name | -collection_name=user_defined_metric_name

[ ] denotes that the parameter is optional
```

Options

- **target_name**

  Internal target type identifier, such as host, oracle_database, and emrep.

- **target_type**

  Name of the target.

- **metric_name**

  Internal name that represents a set of metrics that are collected together. Use the get_on_demand_metrics verb to see the supported list of metrics for a given target.

- **collection_name**

  Name of the user-defined metric or SQL user-defined metric. This option only applies to user-defined metrics and SQL user-defined metrics.

Examples

If you want to collect the "CPU Utilization (%)" metric, look for the appropriate metric internal name (which is Load) using the get_on_demand_metrics command, then run the command as follows:

```
emcli collect_metric  -target_type=host  -target_name=hostname.oracle.com
-metric_name=Load
```

The following example immediately collects and evaluates thresholds for the user-defined metric called MyUDM:

```
emcli collect_metric -target_type=host -target_name=hostname.oracle.com
-collection=MyUDM
```

The following example immediately collects and evaluates thresholds for the SQL user-defined metric called MySQLUDM:

```
emcli collect_metric -target_type=oracle_database -target_name=database
-collection=MySQLUDM
```

confirm_instance

Confirms a manual step.

Format

```
emcli confirm_instance
-instance=[instance_guid]
-stateguid=[state_guids]
```

Options

- **instance**

  Instance GUID.

- **stateguid**

  Comma-separated list of state GUIDs.

Examples

```
emcli confirm_instance -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid=51F762417C4943DEE040578C4E087168
```

```
emcli confirm_instance -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid='51F762417C4943DEE040578C4E087168,51F762417C4944DEE040578C4E087168'
```

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

```
emcli 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"]
      [-is_propagating="true/false"]

[ ] 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.

- **is_propagating**

  Flag that indicates whether or not privilege on the service will be propagated to member targets. The default is false.

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

```
emcli 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]]
         [tzregion:<...>]

[ ] 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`. You can specify this option 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 the 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).

- **schedule=[tzregion:<...>]**

  Time zone region to use. When you "specify" the tzinfo parameter, this parameter determines which timezone to use for the blackout schedule. Otherwise, it is ignored. It defaults to "GMT".

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 America/New_York).

```
emcli create_blackout -name=b1 -add_targets=myhost:host
      -propagate_targets -jobs_allowed
      -schedule="end_time:2007-04-26 05:00;tzinfo:specified;
        tzregion:America/New_York"
      -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 America/New_York). 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;
        tzregion:America/New_York"
      -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 you create the group, you can edit it from the Enterprise Manager Grid Control console to configure Summary Metrics to be displayed for group members.

Format

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

[ ] 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`. You can specify this option more than once.

- **is_propagating**

  Flag that indicates whether or not privilege on the group will be propagated to member targets. The default is false.

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 systems 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="ttype"
        [-settings="setting"]
        [-separator=settings=";"]
        [-subseparator=settings=","]

[ ] 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. Choose one of the following parameters:

  %USERNAME% — Name of the user running the command.
  %RUNAS% — Run the command as this user.
  %COMMAND% — Sudo command.

  The %USER%, %RUNAS%,, %COMMAND% are tokens that the end-user has to use as-is while creating/modifying the privilege delegation settings. The system replaces these token with the actual values at runtime depending the command being run and for which user. Also, %command% should be upper case %COMMAND% for 10.2.0.5 GC.

- **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 (;).

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=","
```

For more examples, see the online help.

create_red_group

Defines a redundancy group name and its members. After you create the redundancy group, you can edit it from the Enterprise Manager Grid Control console to configure charts to be displayed for redundancy group members.

Format

```
emcli 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`. You can specify this option 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_redundancy_group

Creates a redundancy group.

Format

```
emcli create_redundancy_group
       -redundancyGroupName="redGrpName"
       -memberTargetType="tType"
       -memberTargetNames="tName1;tName2"
       [-group_status_criterion="NUMBER" or "PERCENTAGE"]
       [-group_status_tracked="UP" or "DOWN"]
       [-group_status_value=(see the Options section)]
       [-timezone_region=<valid_time_zone_region>]
       [-is_propagating="true/false"]

[ ] denotes that the parameter is optional
```

Options

- **redundancyGroupName**

  Name of the redundancy group.

- **memberTargetType**

  Target type of the constituent member targets.

- **memberTargetNames**

  Member targets for this redundancy group.

- **group_status_criterion**

  This option and the next two calculate the status of the Redundancy Group. Consequently, you need to specify all three options together. If this is not to be a capacity group, you need to specify the following combination:

  ```
  -group_status_criterion='NUMBER' -group_status_tracked='UP' -group_status_
  value='1']
  ```

- **group_status_tracked**

  See the option above.

- **group_status_value**

  See the group_status_criterion option.

  You can specify any value between 1 and 100 if -group_status_criterion= "PERCENTAGE", or any value between 1 and the number of targets present if -group_status_criterion="NUMBER".

- **timezone_region**

  Time zone region of this redundancy group. For a list of valid time zone regions, enter the following command at SQLPLUS:

  ```
  SELECT TZNAME FROM V$TIMEZONE_NAMES
  ```

  You may need to have the SELECT_CATALOG_ROLE role to execute this command.

- **is_propagating**

  Flag that indicates whether or not the privilege on the redundancy group will be propagated to member targets. Possible values are true and false. The default value is false.

Examples

The following example creates a redundancy group with the name 'redGrp1' and with listener, listener2, listener3 as its member targets. The status is calculated as the redundancy group being up if 55 percent of its member targets are up.

```
emcli create_redundancy_group -redundancyGroupName='redGrp1'
      -memberTargetType='oracle_listener'
      -memberTargetNames='listener;listener2;listener3'
      -group_status_criterion='PERCENTAGE'
      -group_status_tracked='UP'
      -group_status_value='55'
```

The following example creates a 'redGrp1' redundancy group with listener, listener2, listener3 as its member targets and time zone as PST8PDT. The status is calculated as the redundancy group being up if 2 of its member targets are up.

```
emcli create_redundancy_group -redundancyGroupName='redGrp1'
          -memberTargetType='oracle_listener'
          -memberTargetNames='listener;listener2;listener3'
          -timezone_region='PST8PDT'
          -group_status_criterion='NUMBER'
          -group_status_tracked='UP'
          -group_status_value='2'
```

create_role

Creates a new Enterprise Manager admininistrator role.

Format

```
emcli 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. You can specify this option more than once.
  Note: Privileges are case-insensitive.

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

  - `CREATE_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`
  - `FULL_TARGET`

  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.example.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.example.com:host"
      -users="johndoe;janedoe"
```

create_service

Creates a service to be monitored by Enterprise Manager.

Format

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

- **hostName**

  Network name of the system 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 you enter the host name, the Agent URL of the host is automatically entered in this field.

- **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'
```

The following example creates a generic service named my service with specified properties with tests defined in mytests.xml, and beacons MyBeacon as the key beacon and MyOtherBeacon as a non-key beacon. Availability is set as test-based.

```
emcli create_service
      -name='my service' -type='generic_service'
      -availType='test' -availOp='or'
      -properties='prop1:value1; prop2:value2'
      -timezone_region='PST8PDT'
      -input_file='template:mytests.xml'
      -beacons='MyBeacon:Y;MyOtherBeacon:N'
```

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

```
emcli 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"]
      [-is_propagating="true/false"]

[ ] 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`. You can specify this option more than once.

- **timezone_region**

  Actual time zone region.

- **owner**

  Owner of the system.

- **meta_ver**

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

- **is_propagating**

  Flag that indicates whether or not the privilege on the system will be propagated to member targets. Possible values are true and false. The default value is false.

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

```
emcli create_user
      -name="name"
      -type="type of user"
      -password="password"
      [-roles="role1;role2;..."]
      [-email="email1;email2;..."]
      [-privilege="name;[[target_name:target_type]|jobid]"]...
      [-profile="profile_name"]
      [-desc="user_description"]
      [-expired="true|false"]
      [-prevent_change_password="true|false"]
      [-input_file="arg_name:file_path"]
```

Options

- **name**

  Administrator name.

- **Type**

  Type of User. The Default value of this parameter is EM_USER. Possible values for this parameter are:

  – EM_USER

  – EXTERNAL_USER

  – DB_EXTERNAL_USER

- **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_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
  - FULL_TARGET

  The following job privileges require specifying jobid:

  - VIEW_JOB
  - FULL_JOB

- **profile**

  Database profile name. It uses DEFAULT as the default profile name.

- **desc**

  User description for the user being added.

- **expired**

  Use this to expire the password immediately. False is the default.

- **prevent_change_password**

  When set to true, you cannot change your own password. False is the default.

- **input_file**

  Allow the administrator to provide the value of any argument in a file. The format of the value will be the name_of_argument:file_path_with_file_name. You can specify this option more than once.

Examples

Example 1

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.example.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.example.com:host"
```

Example 2

The following example makes User1 an Enterprise Manager user, which is already created on an external user store like the SSO server. The contents of priv_file are view_target;host1.example.com:host . User1 will have view privileges on the host1.example.com:host target.

```
emcli create_user
      -name="User1"
      -type="EXTERNAL_USER"
      -input_file="privilege:/home/user1/priv_file"
```

Example 3

The following example makes User1 an Enterprise Manager user, provides a description for the user, and prevents the password from being changed. Only another super administrator could change the password. The profile is set as MGMT_ADMIN_USER_PROFILE.

```
emcli create_user
      -name="User1"
      -desc="This is temp hire."
      -prevent_change_password="true"
      -profile="MGMT_ADMIN_USER_PROFILE"
```

Example 4

The following example makes User1 an Enterprise Manager user, provides a description for the user, and immediately expires the password. When the user logs in the first time, he/she must change the password.

```
emcli create_user
      -name="User1"
      -desc="This is temp hire."
      -expire="true"
```

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

```
emcli 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

Example 1

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

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

Example 2

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_guest_vm

Deletes a guest virtual machine. To delete the guest virtual machine, it should be in the Halted state.

> **Note:** Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:
>
> https://support.oracle.com

Format

```
emcli delete_guest_vm
      -guest_vm_name=Virtual Machine Name
      -server_pool_name=Server Pool Name
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- **server_pool_name**

  Name of the server pool.

Examples

The following example deletes the guest VM dom15, which is in the Oracle Server Pool.

```
emcli delete_guest_vm -guest_vm_name="dom15"
         -server_pool_name="Oracle Server Pool"
```

delete_group

Deletes a group. Deleting a non-existent group generates the error "Group X does not exist."

Format

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

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the group to delete.

- **type**

  Group type: group. Defaults to "group".

Examples

Example 1

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

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

Example 2

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

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

Example 3

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

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

delete_instance

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

```
emcli 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

Example 1

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

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

Example 2

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_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

```
emcli 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

Example 1

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

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

Example 2

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

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

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the service target.

- **type**

  Name of the service type.

- **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.

- **category**

  Defines whether the promoted metric is a usage or a performance metric of a service. This option 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 you specify the category option.

- **promotedMetricColumn**

  Promoted metric column. This is optional if you specify the category option.

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

```
emcli 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

```
emcli 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

```
emcli delete_target
      -name="name"
      -type="type"
      -delete_monitored_targets
```

Options

- **name**

  Target name.

- **type**

  Target type.

- **delete_monitored_targets**

  Delete the targets monitored by the specified the Agent. Applicable only with the oracle_emd 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

```
emcli delete_test
      -name=<target_name>
      -type=<target_type>
      -testname=<test_name>
      -testtype=<test_type>
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **testname**

  Name of the test.

- **testtype**

  Type of test.

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_test_threshold

Deletes a test threshold.

Format

```
emcli delete_test_threshold
      -name=<target_name>
      -type=<target_type>
      -testname=<test_name>
      -testtype=<test_type>
      -metricName=<metric_name>
      -metricColumn=<metric_column>
      [-beaconName=<beacon_name>]
      [-stepName=<step_name>]
      [-stepGroupName=<stepgroupname>]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **testname**

  Test name.

- **testtype**

  Test type.

- **metricName**

  Metric name.

- **metricColumn**

  Metric column.

- **beaconName**

  Beacon name.

- **stepName**

  Step name.

- **stepGroupName**

  Step group name.

Examples

```
emcli delete_test_threshold
         -name="Service Name"
         -type="generic_service"
         -testname="Test Name"
         -testtype="HTTP"
         -metricName="http_response"
         -metricColumn="timing"
```

delete_user

Deletes an existing Enterprise Manager administrator.

When a user is deleted, all 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

```
emcli 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_audit

Disables auditing for all user operations.

Format

```
emcli disable_audit
```

disable_test

Disables monitoring of a Services test.

Format

```
emcli 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'
```

discover_wls

This verb discovers one or more Oracle WebLogic Server Domains. It reads a file labeled domain_discovery_file to discover WebLogic Server versions 7.x, 8.x, 9.x, and 10.x.

domain_discovery_file is required; discovery cannot occur without it. You must create the file prior to performing discovery. To save the discovered components (WebLogic Server versions 9.x and 10.x only) to a specific Management Agent for monitoring, the discover_wls verb reads a second file labeled host_agent_mapping_file. If host_agent_mapping_file does not exist, the Management Agent specified in domain_discovery_file that performs the actual discovery is used as the Agent that will monitor all discovered targets.

Format

```
emcli discover_wls
        -input_file=domain_discovery_file:"domain_discovery_file_
          fully_qualified_path"
        [-input_file=host_agent_mapping_file:"host_agent_mapping_
          fully_qualified_path"]
        [-out_file="output_file_path"]
        [-debug]

[ ] denotes that the parameter is optional
```

Options

- **input_file=domain_discovery_file**

  Fully-qualified path of the CSV (Comma-Separated Values) file that contains multiple lines of domain entries. Each line has the format shown for domain_discovery_file in the "File Structures" section below.

  Note the following points about the format of domain_discovery_file:

  Parameters —

  – <host>,<port>,<username>,<password> are the only mandatory parameters; all others are optional.

  – The order of parameters is fixed. You must provide the parameters in the same order as shown for domain_discovery_file in the "File Structures" section below.

  – If you want to use a comma (,) in any of the parameters provided, you must escape the comma with a backslash as shown in the following example, in which a backslash precedes the comma in the password ihavea,name:

  ```
  10,domain123.xyx.us,11990,weblogic,ihavea\,name,,,farm_demo,<discovery
  agentURL - OPT>
  ```

  Delimeters —

  – Use a comma (,) as the delimiter.

  – The total number of delimiters in each line is fixed and should be equal to 8 (WebLogic Server version 9.x and 10.x) or 9 (WebLogic Server version 7.x and 8.x).

- Delimiters must be present even if the corresponding parameter is not provided. See the last line for domain_discovery_file in the "File Structures" section below.

- **input_file=host_agent_mapping_file**

  Fully-qualified path of the CSV (Comma-Separated Values) file that contains multiple lines of hosts to be monitored and Agents to monitor the hosts. Each line has the following format:

  ```
  <target_host>,<save_to_agent>]
  ```

  Note the following points about the format of host_agent_mapping_file:

  - Use a comma (,) as the delimiter.
  - The total number of delimeters in each line is fixed and should be equal to 1.
  - The order of parameters is fixed. You must provide the parameters in the same order as shown in the sample file structure in the "File Structures" section below. <Discovered Target Host> and <Agent URL to save/monitor the host> are both mandatory parameters.
  - Delimeters must be present even if the corresponding parameter is not provided. See the last line in the sample file structure in the "File Structures" section below.
  - If you want to use a comma (,) in one of the parameters provided, you must escape the comma with a backslash as shown in the following example, in which a backslash precedes the comma in the password ihavea,name:

    ```
    domain123.xyx.us,11990,weblogic,ihavea\,name,,,,,<discovery agentURL - OPT>
    ```

- **out_file**

  Fully-qualified path to a file where you want to redirect the output of this verb.

- **debug**

  Runs this verb in verbose mode for debugging purposes.

File Structures

domain_discovery_file for WebLogic Server versions 7.x and 8.x

The following example shows the structure of a sample domain_discovery_file for WebLogic Server versions 7.x and 8.x. The same Agent is used to discover and save the targets. OPT signifies an optional parameter. The last entry shows the format when the optional parameters, Administration Server Home Directory and Trusted Keystore Filename, are not provided.

```
<WebLogic Server version>,<Administration Server Host>,<port>,<Administration
Server Username>,<password>,<Trusted Keystore Filename - OPT>,<Administration
Server Home Directory - OPT>,<Agent Host>,<Agent Host username>,<Agent Host
password>
<WebLogic Server version>,<Administration Server Host>,<port>,<Administration
Server Username>,<password>,<Trusted Keystore Filename - OPT>,<Administration
Server Home Directory - OPT>,<Agent Host>,<Agent Host username>,<Agent Host
password>
<WebLogic Server version>,<Administration Server Host>,<port>,<Administration
Server Username>,<password>,<Trusted Keystore Filename - OPT>,<Administration
Server Home Directory - OPT>,<Agent Host>,<Agent Host username>,<Agent Host
password>
```

```
<WebLogic Server version>,<Administration Server Host>,<port>,<Administration
Server Username>,<password>,,,<Agent Host>,<Agent Host username>,<Agent Host
password>
```

Definitions for the parameters are as follows for WebLogic Server versions 7 and 8:

- **WebLogic Server version**

  Valid values are 7 or 8. The following example shows a sample entry in domain_discovery_file to discover WebLogic Server version 8:

  ```
  8,myhost.us.mycompany.com,7001,weblogic,welcome1,,,agent_host,agent_
  username,agent_password
  ```

- **Administration Server Host**

  Host name of the WebLogic Administration Server that needs to be discovered. This is a mandatory parameter.

- **port**

  Port of the WebLogic Administration Server.

- **Administration Server Username**

  Login user name for the WebLogic Administration Server.

- password

  Login password for the WebLogic Administration Server.

- Trusted Keystore Filename

  Absolute path of the Trusted Keystore Filename. This is required if the Administration Server's port is SSL enabled.

- Administration Server Home Directory

  Absolute path of the directory where the weblogic.jar file is located.

- Agent Host

  Host name of the Agent used to discover and monitor the discovered targets.

- Agent Host Username | Password

  Credentials of the operating system user of the management Agent host. These credentials are used to discover any Oracle WebLogic Server domains.

domain_discovery_file for WebLogic Server versions 9.x and 10.x

The following example shows the structure of a sample domain_discovery_file for WebLogic Server versions 9.x and 10.x. OPT signifies an optional parameter. The last entry shows the format when optional parameters External Parameters, JMX Protocol, JMX Service URL, and Agent URL are not provided.

```
<WebLogic Server version>,<Administration Server Host>,
<port>,<username>,<password>,<External Parameters - OPT>,<JMX Protocol - OPT>,
<JMX Service URL - OPT>,<Unique Domain Identifier>,<Agent URL -OPT>
<WebLogic Server version>,<Administration Server Host>,
<port>,<username>,<password>,<External Parameters - OPT>,<JMX Protocol - OPT>,
<JMX Service URL - OPT>,<Unique Domain Identifier>,<Agent URL -OPT>
<WebLogic Server version>,<Administration Server Host>,
<port>,<username>,<password>,<External Parameters - OPT>,<JMX Protocol - OPT>,
<JMX    Service URL - OPT>,<Unique Domain Identifier>,<Agent URL -OPT>
<WebLogic Server version>,<Administration Server Host>,
<port>,<username>,<password>,,,,<Unique Domain Identifier>,
```

Definitions for the parameters are as follows for WebLogic Server versions 9 and 10:

- **WebLogic Server version**

  Valid values are 9 or 10. The following example shows a sample entry in domain_discovery_file to discover WebLogic Server version 10:

  ```
  10,myco01.mycompany.com,7001,weblogic,welcome1,,,,soa_farm,
  https://myco01.mycompany.com:3872/emd/main
  ```

- **Administration Server Host**

  Host name of the WebLogic Administration Server that needs to be discovered. This is a mandatory parameter.

- **port**

  Port of the WebLogic Administration Server.

- **username**

  Login user name for the WebLogic Administration Server.

- **password**

  Login password for the WebLogic Administration Server.

- **External Parameters**

  These parameters are passed to the Java process, which connects to the Administration Server. All of these parameters must begin with -D.

- **JMX Protocol**

  Makes a JMX connection to the Administration Server. Valid values are t3, t3s, iiop, and iiops. If you do not provide a protocol, the t3 default is used.

- **JMX Server URL**

  Makes a JMX connection to the Administration Server. If you do not specify this parameter, it is created based on the input parameters.

- **Unique Domain Identifier**

  Creates a unique farm target name. Characters '/', '' are not allowed.

- **Agent URL**

  URL for the Agent used to discover the targets. If you do not provide a value, the local Agent present on the target WebLogic server is used. If an Agent is not found on the target WebLogic Server, an error is displayed.

host_agent_mapping_file

The following example shows the structure of a sample host_agent_mapping_file, valid only for WebLogic Server versions 9.x and 10.x.

```
<Discovered Target Host>,<Agent URL to save/monitor the host>
```

Definitions for the parameters are as follows for WebLogic Server versions 9 and 10:

- **Discovered_Target_Host**

  Any host that has discovered targets.

- **Agent URL to save/monitor the host**

  URL for the Agent to be used to monitor all discovered targets on the corresponding host.

Examples

Example 1

```
emcli discover_wls
       -input_file=domain_discovery_file:/tmp/discovery/emcli/domain_discovery_
        file.csv
```

Example 2

```
emcli discover_wls
       -input_file=domain_discovery_file:/tmp/discovery/emcli/domain_discovery_
        file.csv -debug
```

Example 3

```
emcli discover_wls
       -input_file=domain_discovery_file:/tmp/discovery/emcli/domain_discovery_
         file.csv
       -debug
       -out_file=/tmp/discovery/emcli/logs/emcliLog.txt
```

Example 4

```
emcli discover_wls
       -input_file=domain_discovery_file:/tmp/discovery/emcli/domain_discovery_
        file.csv
       -input_file=host_agent_mapping_file:/tmp/discovery/emcli/host_agent_
        mapping_file.csv
       -debug
       -out_file=/tmp/discovery/emcli/logs/emcliLog.txt
```

enable_audit

Enables auditing for all user operations.

Format

```
emcli enable_audit [-level=basic]

[ ] denotes that the parameter is optional
```

Options

- **-level=basic**

  Enables auditing for `LOGIN`, `LOGOUT`, `DB_LOGIN`, and `DB_LOGOUT`.

Examples

Example 1

The following example enables auditing for all operations:

```
emcli enable_audit
```

Example 2

The following example enables auditing for `LOGIN`, `LOGOUT`, `DB_LOGIN`, and `DB_LOGOUT`:

```
emcli enable_audit [-level=basic]
```

enable_test

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

Format

```
emcli 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

```
emcli 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 you must fully specify the override credential parameters. For host command, `username` and `password` must be specified together.

- **input_file**

  Used in conjunction with the `-osscript` option, this option enables 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

Example 1

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

```
emcli execute_hostcmd
      -cmd="ls -l;"
      -credential_set_name="HostCredsPriv"
      -targets="stach.example.com:host;grp:composite"
```

Example 2

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.example.com:host` and host targets contained in the group `grp`. The stored `HostCredsNormal` preferred credentials are 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.example.com:host;grp:composite"
```

execute_sql

Executes a SQL command across a set of targets.

Format

```
emcli 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

Example 1

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 are used for all the targets.

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

Example 2

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 are 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_masking_definition

Exports a masking definition in XML format.

Format

```
emcli export_masking_definition
     -definition_name=<masking_definition_name>
     [-path=file path]
     [-file=file name]
```

Options

- **definition name**

  Masking definition name.

- **path**

  Path for the file name to save the masking script. The file name is auto generated. -path and -file are mutually exclusive. Only an absolute path is allowed.

- **file**

  File name to save the masking script. The file name must include the absolute path. -path and -file are mutually exclusive

Output Columns

Success/Error messages.

Examples

Example 1

The following example exports the masking definition mask_hr_data to an XML file at the specified path:

```
emcli export_masking_definition
      -definition_name=mask_hr_data
      -path=/tmp/
```

Example 2

The following example exports the masking definition mask_hr_data to an XML file named abc.xml:

```
emcli export_masking_definition
      -definition_name=mask_hr_data
      -file=/tmp/abc.xml
```

export_report

Exports a report definition and all of its element definitions given its title and owner.

Format

```
emcli export_report
  -title="<report-title>"
  -owner="<report-owner>"
  -output_file="<file>"
```

Options

- **title**

  Title of the report to export.

- **owner**

  The owner of the report to export. The logged-in emcli user must have view privilege for the report. Target names are not exported. The report is uniquely defined using title and owner, so both must be supplied.

- **output_file**

  Name of the exported file.

Examples

```
emcli export_report
  -title="maintenance report"
  -owner="SHIFT1_OPERATOR"
  -output_file="$HOME/reports/maint_report.xml"
```

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

Example 1

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
```

Example 2

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. This is 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**

  Runs 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.

Examples

```
emcli extend_as_home
   -input_file="dest_properties:/home/destinations.txt"
   -list_exclude_files="centralagents.lst"
   -isSwLib="false"
   -tryftp_copy="false"
   -jobname="extend as home"
   -iasInstance="asinstancename"
   -isIas1013="false"
   -clustername=ascluster
   -oldIASAdminPassword="oldpassword"
   -newIASAdminPassword="newpassword"
   -prescripts="/home/abc/myscripts"
   -run_prescripts_as_root="true"
   -rootscripts="%oracle_home%/root.sh"
   -source_params="TargetName:host.domain.com;HomeLoc=/home/oracle/appserver1;
    HomeName=oracleAppServer1;ScratchLoc=/tmp"
```

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"
   -clustername="name of cluster to create"
   -isSwLib="true/false"
   -tryftp_copy="true/false"
   -jobname="name of cloning job"
    [-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.

- **clustername**

  Name of the cluster to create.

- **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.

- **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;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 `%%`.

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

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

[ ] denotes that the parameter is optional
```

Options

- **templateName**

  Name of the template.

- **templateType**

  Type of template.

- **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**

  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'
```

Note:

- The emcli user must have operator privilege on the repository template to perform this operation.
- Beacon-related information is not exported to the file. In particular, the list of monitoring beacons, as well as any beacon-specific properties or thresholds, are not exported.
- The values of password variables are not exported.

generate_masking_script

Generates a masking script for the given masking definition.

Format

```
 emcli generate_masking_script
-definition_name=masking definition name
[-parameters="name1:value1;name2:value2;..."]
[-credential_set_name=credential_set_name]
[-input_file="parameter_tag:file_path"]
[-script | -format=[name:<pretty|script|csv>];
           [column_separator:column_sep_string];
           [row_separator:row_sep_string];
]
```

Options

- **definition_name**

  Name of the masking definition.

- **parameters**

  List of name-value pairs that represent the credentials required for connecting to the database instance. The supported parameters are db_username, db_password, and db_role.

- **credential_set_name**

  Set name of the preferred credentials stored in the Enterprise Manager repository. The valid values for this parameter are:

  – **DBCredsNormal** —

    Default normal credential set for an oracle_database target. If no value is specified, DBCredsNormal is used.

  – **DBCredsSYSDBA** —

    SYSDBA credential set for an oracle_database target.

  You can only specify this parameter when the override credential parameters such as db_username and db_password are not present.

- **input_file**

  Used in conjunction with the 'parameters' option, this option enables you to store parameter values, such as username and password, in a separate file. This option specifies a mapping between a tag and a local file path. The tag is specified in lieu of specific parameter values of the 'parameters' option. The tag must not contain colons (:) or semi-colons (;).

- **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.
- format="name:script;column_separator:<column_sep_string>" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
- format="name:script;row_separator:<row_sep_string>" row-separates the Verb output by <row_sep_string>. Columns are separated by the tab character.

Output

Success or error messages as well as the impact report (if generated).

Examples

Example 1

The following example generates a script for the masking definition named mask_hr_data :

```
emcli generate_masking_script
        -definition_name=mask_hr_data
        -parameters="db_username:system;db_password:password;db_role:NORMAL"
```

Example 2

The following example generates a script for the masking definition named mask_hr_data. The database password is read from the pwd.txt file.

```
emcli generate_masking_script
-definition_name=mask_hr_data
-parameters=PWD_FILE
-input_file="PWD_FILE:pwd.txt"
```

Example 3

The following example reads the credentials from the preferred credential set DBCredsNormal and generates the masking script.

```
emcli generate_masking_script
-definition_name=mask_hr_data
```

Example 4

The following example reads the credentials from the preferred credential set DBCredsSYSDBA and generates the masking script.

```
emcli generate_masking_script
-definition_name=mask_hr_data
-credential_set_name=DBCredsSYSDBA
```

get_agent_properties

Displays Agent properties. You can use this command if you have view privilege for the Agent.

Format

```
emcli get_agent_properties
      -agent_name="<agent_target_name>"
      [-all]
      [-format="<format_name>"]

[ ] denotes that the parameter is optional
```

Options

- **agent_name**

  Name of the Agent target.

- all

  Shows all Agent properties. By default, only basic properties appear.

- **format**

  Format to display Agent properties. Valid values are pretty, script, and csv. By default, values are displayed in pretty format.

Examples

The following example shows all of the Agent properties in CSV format:

```
emcli get_agent_properties -agent_name="agent.example.com:11850"
        -all
        -format=csv
```

get_agent_property

Displays the value of a specific Agent property. You can use this command if you have view privilege for the Agent.

Format

```
emcli get_agent_property
      -agent_name="<agent_target_name>"
      -name="<agent_property_name>"
```

Options

- **agent_name**

  Name of the Agent target.

- name

  Name of the Agent property.

Examples

The following example shows the current value of the UploadInterval property in emd.properties.

```
emcli get_agent_property -agent_name="agent.example.com:11850"
         -name=UploadInterval
```

get_aggregate_service_info

Gets time zone and availability evaluation function information of an aggregate's service instance.

Format

```
emcli 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.

- **noheader**

  Displays 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.
  - `format=`column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

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

```
emcli 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.

- **noheader**

  Displays 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.
  - `format=`column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Examples

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

get_blackout_details

Gets detailed information for a specified blackout.

Format

```
emcli 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**

  Displays 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.
  - format="name:script;column_separator:<column_sep_string>" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - format="name:script;row_separator:<row_sep_string>" row-separates the Verb output by <row_sep_string>. Columns are separated by the tab character.

Output Columns

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

Examples

Example 1

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

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

Example 2

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

```
emcli 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

```
emcli 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**

  Displays 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.
  - format="name:script;column_separator:<column_sep_string>" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - format="name:script;row_separator:<row_sep_string>" row-separates the Verb output by <row_sep_string>. Columns are separated by the tab character.

Output Columns

Target Name, Target Type, Status, Status ID

Examples

Example 1

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

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

Example 2

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

```
emcli 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**

  Lists 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**

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

- **noheader**

  Displays 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.

  – format="name:script;column_separator:<column_sep_string>" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

  – format="name:script;row_separator:<row_sep_string>" row-separates the Verb output by <row_sep_string>. Columns are separated by the tab character.

Output Columns

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

Examples

Example 1

The following example shows all blackouts with some details.

```
emcli get_blackouts
```

Example 2

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

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

Example 3

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

```
emcli get_blackouts -hostnames=myhost.example.com
```

Example 4

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

```
emcli get_blackouts -hostnames="myhost.example.com"
-hostnames="yourhost.example.com"
```

get_ca_info

Displays information about all of the Certificate Authorities (CA) created since the Grid Control installation. It also displays the Agent names whose certificates are issued by the CA(s) when you specify the -details option. The following information is retrieved from the Grid Control repository:

- Unique identifier of the Certificate Authority (CA) in the Grid Control repository
- CA description
- CA creation date
- CA expiration date
- Number of Agents registered to this CA
- Number of secured Agents not registered to any CA

Format

```
emcli get_ca_info
        [-ca_id="id1;id2;..."] [-details]

[ ] denotes that the parameter is optional
```

Options

- **ca_id**

  Specifies the Certificate Authority ID.

- **details**

  For each Certificate Authority, displays the list of Agent names whose certificates are issued by it.

Examples

The following example shows output for for the CA with the ID of 2 specified.

```
emcli get_ca_info -ca_id=2

 Info about CA with ID: 2
 CA is configured
 DN: EMAILADDRESS=Enterprise.Manager@myomshost.mycompany.com,
 CN=myomshost.mycompany.com, OU=EnterpriseManager on myomshost.mycompany.com,
 O=EnterpriseManager on myomshost.mycompany.com, L=EnterpriseManager on
 myomshost.mycompany.com1, ST=CA, C=US, DC=com
 Serial# : 87539237298512593900
 Valid From: Mon Oct 25 17:01:15 UTC 2010
 Valid Till: Thu Oct 22 17:01:12 UTC 2020
 Number of Agents registered with CA ID 2 is 1

 Number of Agents to be re-secured, as OMS is secured using force_newca
    option: 1
```

Regarding the `force_newca` option in the last line, the output shows that a new certificate was created with the ID of 2. Two agents have been re-secured to be registered with this new certificate. The OMS running on `myomshost.mycompany.com` has been re-secured to be registered with the new certificate created. The command "`emctl secure oms`" failed because a secured

Agent was not registered to the new CA, hence the command "`emctl secure oms force-newca`" to secure the OMS anyway. There is still an Agent that needs to be secured to be registered to the new certificate. To retrieve the Agent name, you need to run the command "`emcli get_ca_info -ca_id=2 -details`," which is shown in the next example.

The following example displays the Agent names registered with the CA(s) for ID 2.

```
emcli get_ca_info -ca_id=2 -details

     Info about CA with ID: 2
     CA is configured
     DN: EMAILADDRESS=Enterprise.Manager@myomshost.mycompany.com,
      CN=myomshost.mycompany.com, OU=EnterpriseManager on myomshost.mycompany.com,
       O=EnterpriseManager on myomshost.mycompany.com, L=EnterpriseManager on
       myomshost.mycompany.com2, ST=CA, C=US, DC=com
     Serial# : 87539237298512593900
     Valid From: Mon Oct 25 17:01:15 UTC 2010
     Valid Till: Thu Oct 22 17:01:12 UTC 2020
     Number of Agents registered with CA ID 2 is 1
     usagent1.mycompany.com:20872

    Following Agents needs to be re-secured, as OMS is secured using force_newca
     option:

     ukagent1.mycompany.com:1830
```

get_guest_vm_status

Gets the status of the guest virtual machine.

> **Note:** Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:
>
> https://support.oracle.com

Format

```
emcli get_guest_vm_status
      -guest_vm_name=<Guest VM Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- **server_pool_name**

  Name of the server pool.

Output Column

Status

Examples

The following example gets the guest status for VM st-vs1.example.com.

```
emcli get_guest_vm_status -guest_vm_name="st-vs1.example.com"
 -server_pool_name="Oracle Server Pool"
```

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

```
emcli get_group_members
      -name="name"
      [-type=<group>]
      [-depth=#  (default 1)]
      [-noheader]
      [-expand_non_groups]
      [-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**

  Lists 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 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**

  Displays tabular information without column headers.

- **expand_non_groups**

  Lists members of aggregates and the aggregate target. By default, only sub-group target members are listed.

- **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.
- `format=`column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
- row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Output Columns

Target Name, Target Type

Examples

Example 1

The following example lists the databases in group `db2_group`.

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

Example 2

The following example verifies that group `my_hosts:group` exists.

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

Example 3

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
```

Example 4

The following example lists the unique targets in group my_group:group and its sub-groups/aggregates. The aggregate targets are also listed.

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

get_groups

Lists all groups.

Format

```
emcli 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**

  Displays 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.
  - `format=`column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

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 generates 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_instance_status

Displays the procedure instance status identified by the GUID on the command line.

> **Tip:** See also get_instances on page 2-128 and get_job_execution_detail on page 2-129.

Format

```
emcli get_instance_status -guid=<guid_number>
[-xml [-details] [-showJobOutput [-tailLength=<Last N Characters>]]]
```

Options

- **guid**

  Displays the details of a procedure instance identified by the GUID number. You can find this number by executing the get_instances command.

- **xml**

  Shows the complete status of each of the steps in XML format.

- **details**

  Displays more details for the command output. This option also requires the -xml option.

- **showJobOutput**

  Shows the output or errors for the job execution steps. This option also requires the -xml option.

- **tailLength**

  Limits the number of characters in the job step output or error. This option also requires the -showJobOutput option.

  <Last N Characters> is a positive non-zero number until which the characters are chosen from the end of the job step output. The system sets the maximum permissible characters to dump. If you do not provide this option, the maximum permissible characters are dumped.

Output Columns

GUID, Procedure Type, Instance Name, Status

Examples

Example 1

The following example shows procedure details in CSV format:

```
emcli get_instance_status -guid=12345678901234567890123456789012
```

Example 2

The following example shows details in XML format:

```
emcli get_instance_status -guid=16B15CB29C3F9E6CE040578C96093F61 -xml -details
```

Example 3

The following example shows details in XML format with complete output:

```
emcli get_instance_status -guid=16B15CB29C3F9E6CE040578C96093F61 -xml -details
-showJobOutput
```

Example 4

The following example shows details in XML format with the last 1024 characters of output:

```
 emcli get_instance_status -guid=16B15CB29C3F9E6CE040578C96093F61 -xml
-showJobOutput -tailLength=1024
```

get_instances

Displays a list of procedure instances.

Tip: See also get_procedure_types on page 2-134.

Format

```
emcli get_instances [-type=<procedure_type>]

[  ] denotes that the parameter is optional
```

Options

- **type**

  Displays all the procedure instances of type procedure_type.

Output Columns

GUID, Procedure Type, Instance Name, Status

Examples

Example 1

The following example lists all procedure instances:

```
emcli get_instances
```

Example 2

The following example lists all procedure instances of type 'PatchOracleSoftware':

```
emcli get_instances -type=PatchOracleSoftware
```

get_job_execution_detail

Displays details of a job execution.

Format

```
emcli get_job_execution_detail
       -execution=<execution_id>
       [-xml [-showOutput [-tailLength=<length>]]]

[ ] denotes that the parameter is optional
```

Options

- **execution**

  Specifies that the ID of the job execution (execution_id) is the job execution ID.

- **xml**

  Shows the execution details as XML.

- **showOutput**

  Shows the output of the steps inside the job execution. You can only use this option in conjunction with the -xml option.

- **tailLength**

  Limits the display of the output to the number of characters from the end of the output. (length) is in characters. You can only use this option in conjunction with the -showOutput option. If you do not specify this option, a system-generated hard limit is enforced.

Examples

Example 1

The following example shows the details in CSV format:

```
emcli get_job_execution_detail -execution=12345678901234567890123456789012345
```

Example 2

The following example shows the details in XML format:

```
emcli get_job_execution_detail -execution=12345678901234567890123456789012 -xml
```

Example 3

The following example shows the details in XML format with complete output:

```
emcli get_job_execution_detail -execution=12345678901234567890123456789012 -xml
-showOutput
```

Example 4

The following example shows the details in XML format with last N chars output:

```
emcli get_job_execution_detail -execution=12345678901234567890123456789012 -xml
-showOutput -tailLength=1024
```

get_jobs

Lists existing jobs.

Format

```
emcli 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**

  Lists job IDs to use as the output filters.

- **targets**

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

- **status_ids**

  Lists 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**

  Displays 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.

  – `format`=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

  – row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Output Columns

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

Examples

Example 1

The following example shows the jobs with the specified job IDs 12345678901234567890123456789012 and 09876543210987654321098765432100:

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

Example 2

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

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

Example 3

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_metrics_for_stateless_alerts

For the specified target type, lists the metrics whose alerts are stateless and thus can be manually cleared. Both the metric name and metric internal name are provided in the output of this command. To clear the stateless alerts associated with the specified metric, use the clear_stateless_alerts verb.

Format

```
emcli get_metrics_for_stateless_alerts
      -target_type=type
```

Options

- **target_type**

  Internal target type identifier, such as host, oracle_database, oc4j, oracle_emrep, and oracle_emd.

Examples

The following example provides a list of all metrics for which stateless alerts can be manually cleared for any Oracle database (internal name for the target type is oracle_database).

```
emcli get_metrics_for_stateless_alerts -target_type=oracle_database
```

get_on _demand_metrics

Gets a list of metrics that can be immediately collected with the collect_metric EMCLI verb. From this list, identify the metric you are interested in under the Metric Name column, then use its corresponding Metric Internal name in the collect_metric verb.

Format

```
emcli get_on_demand_metrics
      -target_type=type
      -target_name=name
```

Options

- **target_type**

  Internal target type identifier, such as host, oracle_database, oc4j, oracle_emrep, and oracle_emd.

- **target_name**

  Name of the target.

Examples

The following example shows a list of collectible metrics for the host target called hostname.oracle.com.

```
emcli get_on_demand_metrics -target_type=host -target_name=hostname.oracle.com
```

get_procedure_types

Gets the list of all Deployment Procedure types.

Format

```
emcli get_procedure_types
```

Output Column

Procedure Type

Examples

The following example lists all procedure types:

```
emcli get_procedure_types
```

get_procedure_xml

Gets the Deployment Procedure XML file. XML is 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.

Tip: See also get_procedure_types on page 2-283.

Format

```
emcli get_procedures [-type=<procedure_type>]

[ ] denotes that the parameter is optional
```

Options

- **type**

  Display all the Deployment Procedures of type procedure_type.

Output Columns

GUID, Procedure Type, Name, Version, Created By

Examples

Example 1

The following example lists all procedures:

```
emcli get_procedures
```

Example 2

The following example lists all procedures of type 'AS Provisioning':

```
emcli get_procedures -type="AS Provisioning"
```

get_reports

Returns a list of reports owned by or viewable by all users or a specified user. The output of this report is space-separated, quoted strings for the report title and owner, with each report on its own line.

Format

```
emcli get_reports
   [-owner="<report-owner>"]

[ ] denotes that the parameter is optional
```

Options

- **owner**

  Enables listing of viewable reports that a specific Enterprise Manager owns.

Examples

```
emcli get_reports -owner=username
"report 1","username"
"example report 2","username"


emcli get_reports
"report A","username1"
"report 1","username2"
"example report 2","username2
```

get_retry_arguments

Get arguments of failed steps that can be retried.

Format

```
emcli get_retry_arguments
-instance=<instance_guid>
[-stateguid=<state_guid>]

[ ] denotes that the parameter is optional
```

Options

- **instance**

  Instance GUID.

- **stateguid**

  State GUID.

Examples

```
emcli get_retry_arguments -instance=16B15CB29C3F9E6CE040578C96093F61
```

```
emcli get_retry_arguments -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid=51F762417C4943DEE040578C4E087168
```

get_system_members

Lists the members of the specified system.

Format

```
emcli 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**

  Lists target members in sub-systems to the specified depth. 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**

  Displays 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.

  – `format=`column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

  – row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Output Columns

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

Examples

Example 1

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

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

Example 2

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

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

Example 3

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_target_properties

Lists all the property names for the target type provided.

Format

```
emcli get_target_properties
         -target_type="target_type"
```

Options

- **target_type**

  Target type for which you want to list user-defined property names.

Examples

```
emcli get_target_properties -target_type="host"

Comment
Contact
Deployment Type
Line of Business
Location
Target properties fetched successfully
```

get_targets

Gets status and alert information for targets.

Format

```
emcli 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.
  - `format=`column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Output Columns

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

Examples

Example 1

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

```
emcli get_targets
```

Example 2

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

```
emcli get_targets
      -alerts
```

Example 3

The following example shows all `oracle_database` targets.

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

Example 4

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

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

Example 5

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

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

Example 6

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

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

get_test_thresholds

Shows test thresholds.

Format

```
emcli get_test_thresholds
     -name=<target_name>
     -type=<target_type>
     -testname=<test_name>
     -testtype=<test_type>
     [-script|-format=
             [name:"pretty|script|csv"];
             [column_separator:"sep_string"];
             [row_separator:"row_sep_string"]
     ]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Target name.

- **type**

  Target type.

- **testname**

  Test name.

- **testtype**

  Test type.

- **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.

  – `format`=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

  – row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Examples

```
emcli get_test_thresholds -name="Service Name"
                -type="generic_service"
                -testname="Test Name"
                -testtype="HTTP"
```

get_unsync_alerts

Gets a list of alerts that are out-of-sync between the Agent and the repository for the specified target. You would typically use this command when you think that the Agent has not uploaded the latest alert to the repository. Under these circumstances, the repository would be out-of-sync with the Agent state.

Format

```
emcli get_unsync_alerts
      -target_type="type"
      -target_name="name"
```

Options

- target_type

  Internal target type identifier, such as host, oracle_database, emrep, and so forth.

- **target_name**

  Name of the target.

Output Column

Status

Examples

The following example shows the out-of-sync alert states for the host target type and abc.oracle.com target name:

```
emcli get_unsync_alerts -target_type=host -target_name=abc.oracle.com
```

get_virtual_server_status

Gets the status of the virtual server.

> **Note:** Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:
>
> https://support.oracle.com

Format

```
emcli get_virtual_server_status
         -server_name=<Virtual Server Name>
```

Options

- **server_name**

  Name of the virtual server.

Output Column

Status

Examples

The following example gets the status of the virtual server st-vs1.example.com.

```
emcli get_virtual_server_status -server_name="st-vs1.example.com"
```

grant_license_no_validation

Grants licenses on a set of user-specified packs, or all packs to a set of user-specified targets, or all targets belonging to the input licensable target type.

For 11g database targets, you cannot enable or disable the Database Diagnostic and Tuning Packs through the user interface. You need to set the control_management_pack_access initialization parameter to manage your licenses. For information about this parameter, see the Enterprise Database Management chapter of *Oracle Enterprise Manager Licensing Information*.

> **Tip:** You can use this verb to grant licenses for standalone target types, such as hosts and databases, but you cannot use this verb to grant licenses for the parent Application Server (oracle_ias) target type, which has dependent target types of OC4J, Jserv, Web Cache, and so forth. To do this, use the grant_license_with_validation verb instead.
>
> For example, for pack ias_config and an Application Server target of AS1 with an associated dependent target of OC4J1, this verb grants a license to AS1, but this does not propagate to OC4J1.

Format

```
emcli grant_license_no_validation
      -type="target_type"
      [-targets="tname1;tname2;..."]
      [-packs="pack1;pack2;..."]
      [-file="file_name"]
      [-displayAllMessages]

[ ] denotes that the parameter is optional
```

Options

- **type**

  Target type as it exists in the database. Names cannot contain colons (:), semi-colons (;), or any leading or trailing blanks. You can specify only one target type at a time; for example, -type="oracle_database".

- **targets**

  Targets should be specified in the following sequence:

  ```
  TargetName1;TargetName2;
  ```

  For example:

  ```
  -targets="database1;database2;database3;"
  ```

  The semi-colon (;) is the target separator.

  See the "Examples" section below for information about providing arguments for the targets option.

- **packs**

  License packs should be specified in the following sequence:

  ```
  pack1;pack2;
  ```

For example:

```
-packs="db_diag;db_config;"
```

The semi-colon (;) is the pack separator.

See the "Examples" section below for information about providing arguments for the packs option.

- **file**

  Specify the file name, including the complete path. For example:

  ```
  -file="/usr/admin1/db_license.txt"
  ```

  The file should contain the list of targets and packs according to the following cases:

  – If you only need to provide a list of targets, use the following format:

    ```
    targets=database1;database2;database3;
    ```

  – If you only need to provide a list of packs, use the following format:

    ```
    packs=db_diag;db_config;
    ```

  – If you need to provide a list of both targets and packs, use the following format:

    ```
    targets=database1;database2;database3;
    packs=db_diag;db_config;
    ```

- **displayAllMessages**

  Displays all messages. Only error messages are displayed by default. "=value" is not allowed on the command line.

Examples

Example 1 and Example 2 below grant licenses to specific packs for specific targets. In order to know which target types and pack names you can pass as arguments, you can use the view named mgmt_license_view to see a list of licensable targets, their target types, and the list of packs licensed on them.

To obtain this information, do the following:

1. Access SQL*Plus with your username and password, using sysman or other user that has access to sysman.mgmt_license_view.
2. Select a distinct pack name from sysman.mgmt_license_view, where:

   ```
   target_type=<oracle_database>
   ```

The following example shows pack names for an Oracle database you specify as the target type.

```
PACK_NAME
------------------
db_config
provisioning
db_sadm
db_tuning
db_diag
provisioning_db
db_chgmgt
```

```
7 rows selected.
```

Based on this information, to grant a license to the database1 target for the db_chgmgt pack, you would enter the following command:

```
emcli grant_license_no_validation -type="oracle_database" -targets="database1"
-packs="db_chgmgt"
```

The only limitation of mgmt_license_view is that it only lists the packs for a target type where the pack is granted to at least one target of that type. That is, if the pack is not granted to any target of that type, mgmt_license_view cannot provide any information.

Example 1

The following example grants the license to the db_diag and db_config packs to database1, database2, and database3 targets (oracle_database target type):

```
emcli grant_license_no_validation -type="oracle_database"
          -targets="database1;database2;database3;" -packs="db_diag;db_config;"
```

Example 2

The following example grants the license to the db_diag and db_config packs to all database targets in the setup:

```
emcli grant_license_no_validation -type="oracle_database"
       -packs="db_diag;db_config;"
```

Example 3

The following example grants the license to all packs (applicable to database targets) to database1, database2, and database3 targets in the setup:

```
emcli grant_license_no_validation -type="oracle_database"
         -targets="database1;database2;database3;"
```

Example 4

The following example grants the license to all packs (applicable to database targets) to all database targets in the setup:

```
emcli grant_license_no_validation -type="oracle_database"
```

Example 5

The following example uses a text file to pass targets and pack names as the argument. It grants the license to the db_diag and db_config packs to the database1, database2, and database3 targets (oracle_database target type):

```
emcli grant_license_no_validation -type="oracle_database"
          -file="/usr/admin1/db_license.txt"
             targets=database1;database2;database3;
                   packs=db_diag;db_config;
```

... where the content of the "/usr/admin1/license/db_license.txt" file is as follows:

```
targets=database1;database2;database3;
packs=db_diag;db_config;
```

grant_license_with_validation

Grants licenses on a set of user-specified packs, or all packs to a set of user-specified targets, or all targets belonging to the input licensable target type as per business rules.

For 11g database targets, you cannot enable or disable the Database Diagnostic and Tuning Packs through the user interface. You need to set the control_management_pack_access initialization parameter to manage your licenses. For information about this parameter, see the Enterprise Database Management chapter of *Oracle Enterprise Manager Licensing Information*.

> **Tip:** You can use this verb to grant licenses for standalone target types, such as hosts and databases, and you also use this verb to grant licenses for the parent Application Server (oracle_ias) target type, which has dependent target types of OC4J, Jserv, Web Cache, and so forth.
>
> For example, for pack ias_config and an Application Server target of AS1 with an associated dependent target of OC4J1, this verb grants a license to AS1 and also propagates to OC4J1 (and all other dependent targets associated with AS1).
>
> To grant licenses for only standalone target types, use the grant_license_no_validation verb.

Format

```
emcli grant_license_with_validation
      -type="target_type"
      [-targets="tname1;tname2;..."]
      [-packs="pack1;pack2;..."]
      [-file="file_name"]
      [-displayAllMessages]

[ ] denotes that the parameter is optional
```

Options

- **type**

  Target type as it exists in the database. Names cannot contain colons (:), semi-colons (;), or any leading or trailing blanks. You can specify only one target type at a time; for example, -type="oracle_database".

- **targets**

  Targets should be specified in the following sequence:

  ```
  TargetName1;TargetName2;
  ```

  For example:

  ```
  -targets="database1;database2;database3;"
  ```

  The semi-colon (;) is the target separator.

  See the "Examples" section below for information about providing arguments for the targets option.

- **packs**

  License packs should be specified in the following sequence:

```
pack1;pack2;
```

For example:

```
-packs="db_diag;db_config;"
```

The semi-colon (;) is the pack separator.

See the "Examples" section below for information about providing arguments for the packs option.

- **file**

  Specify the file name, including the complete path. For example:

  ```
  -file="/usr/admin1/db_license.txt"
  ```

  The file should contain the list of targets and packs according to the following cases:

  – If you only need to provide a list of targets, use the following format:

    ```
    targets=database1;database2;database3;
    ```

  – If you only need to provide a list of packs, use the following format:

    ```
    packs=db_diag;db_config;
    ```

  – If you need to provide a list of both targets and packs, use the following format:

    ```
    targets=database1;database2;database3;
    packs=db_diag;db_config;
    ```

- **displayAllMessages**

  Displays all messages. Only error messages are displayed by default. "=value" is not allowed on the cmd line.

Examples

Example 1 and Example 2 below grant licenses to specific packs for specific targets. In order to know which target types and pack names you can pass as arguments, you can use the view named mgmt_license_view to see a list of licensable targets, their target types, and the list of packs licensed on them.

To obtain this information, do the following:

1. Access SQL*Plus with your username and password, using sysman or other user that has access to sysman.mgmt_license_view.
2. Select a distinct pack name from sysman.mgmt_license_view, where:

   ```
   target_type=<oracle_database>
   ```

The following example shows pack names for an Oracle database you specify as the target type.

```
PACK_NAME
-------------------
db_config
provisioning
db_sadm
db_tuning
db_diag
```

```
provisioning_db
db_chgmgt

7 rows selected.
```

Based on this information, to grant a license to the database1 target for the db_chgmgt pack, you would enter the following command:

```
emcli grant_license_with_validation -type="oracle_database" -targets="database1"
-packs="db_chgmgt"
```

The only limitation of mgmt_license_view is that it only lists the packs for a target type where the pack is granted to at least one target of that type. That is, if the pack is not granted to any target of that type, mgmt_license_view cannot provide any information.

Example 1

The following example grants a license to the db_diag and db_config packs to database1, database2, and database3 targets (oracle_database target type):

```
emcli grant_license_with_validation -type="oracle_database"
          -targets="database1;database2;database3;" -packs="db_diag;db_config;"
```

Example 2

The following example grants a license to the db_diag and db_config packs to all database targets in the setup:

```
emcli grant_license_with_validation -type="oracle_database"
       -packs="db_diag;db_config;"
```

Example 3

The following example grants a license to all packs (applicable to database targets) to database1, database2, and database3 targets in the setup:

```
emcli grant_license_with_validation -type="oracle_database"
          -targets="database1;database2;database3;"
```

Example 4

The following example grants a license to all packs (applicable to database targets) to all database targets in the setup:

```
emcli grant_license_with_validation -type="oracle_database"
```

Example 5

The following example uses a text file to pass targets and pack names as the argument. It grants a license to the db_diag and db_config packs to the database1, database2, and database3 targets (oracle_database target type):

```
emcli grant_license_with_validation -type="oracle_database"
          -file="/usr/admin1/db_license.txt"
             targets=database1;database2;database3;
                   packs=db_diag;db_config;
```

where the content of the "/usr/admin1/license/db_license.txt" file is as follows:

```
targets=database1;database2;database3;
packs=db_diag;db_config;
```

grant_privs

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

> **Note:** To replace an existing Enterprise Manager administrator role, use the modify_role verb.

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:

  `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 job privileges require specifying `jobid`:

  `VIEW_JOB`

  `FULL_JOB`

Examples

Example 1

The following example grants these privileges to user1:

- Privilege to use any beacon
- Full control of the jobs with ID 923470234ABCDFE23018494753091111
- Full control on the target host1.example.com:host

```
emcli grant_privs
          -name="user1"
          -privilege="USE_ANY_BEACON"
```

```
          -privilege="FULL_JOB;923470234ABCDFE23018494753091111"
          -privilege="FULL_TARGET;host1.example.com:host"
```

Example 2

The following example grants target privileges to EM Role : Role1:

```
emcli grant_privs
          -name="Role1"
          -privilege="FULL_TARGET;host1.example.com:host"
```

grant_roles

Grants 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

```
emcli grant_roles
      -name="user1"
      -roles="SUPER_USER"

emcli grant_roles
      -name="Role1"
      -roles="BLACKOUT_ADMIN;MAINTAIN_TARGET"
```

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

```
emcli help [verbname]

[ ] denotes that the parameter is optional
```

Options

None.

Examples

Example 1

The following example provides an overview for all available verbs:

```
emcli help
```

Example 2

The following example provides the description, syntax, and usage examples for the add_target verb:

```
emcli help add_target
```

ignore_instance

Ignores a failed step.

Format

```
emcli ignore_instance
-instance=<instance_guid>
[-stateguid=<state_guid>]

[ ] denotes that the parameter is optional
```

Options

- **instance**

  Instance GUID.

- **stateguid**

  Comma-separated list of state GUIDs.

Examples

```
emcli ignore_instance -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid=51F762417C4943DEE040578C4E087168


emcli ignore_instance -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid='51F762417C4943DEE040578C4E087168,51F762417C4944DEE040578C4E087168'
```

import_masking_definition

Imports a masking definition from the specified XML file.

Format

```
emcli import_masking_definition
     -file=/tmp/file_name.xml
```

Options

- **file**

  Path of the file containing the masking definition in XML format.

Output

Success or error messages

Examples

The following example imports the masking definition from the hr_mask.xml file.

```
emcli import_masking_definition
       -file=/tmp/hr_mask.xml
```

import_report

Imports one or more report definitions from an XML file(s) using the title in the XML file and the currently logged-in CLI user as the owner of the report. If the report/owner already exists, the operation fails for this report with an accompanying error message. (You can override this with the optional -force option.) The report will be changed to a just-in-time report with the target type from the exported report.

You will need to edit schedules and access privileges using the UI. The system enforces title/owner uniqueness, so an error occurs if a report with the same title and owner already exists.

Format

```
emcli import_report
   [-force]
   -files="file1;file2;..."
```

Options

- **force**

  First delete the report (and all jobs and saved copies) if a report with the same title/owner exists.

- **files**

  List of path/file name(s) of XML file(s) that contain valid report definition(s).

Examples

```
emcli import_report \
     -files="$HOME/reports/maint_report1.xml;$HOME/reports/file2.xml"
```

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

Example 1

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

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

Example 2

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

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

list_guest_vm

Lists all guest virtual machines.

> **Note:** Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:
>
> https://support.oracle.com

Format

```
emcli list_guest_vm
          [-server_pool_name=<Server Pool Name>]
          [-server_name=<Virtual Server Name>]
          [-operating_system=<Operating system name>]

[ ] denotes that the parameter is optional
```

Options

- **server_pool_name**

  Name of the server pool.

- **server_name**

  Name of the virtual server.

- **operating_system**

  Guest VM's operating system name.

Output Columns

Name, Status, Operating system, VNC url, Server name, Server pool name

Examples

```
emcli list_guest_vm  -server_pool_name="Oracle Server Pool"
emcli list_guest_vm  -server_name="st-vs1.oracle.com"
emcli list_guest_vm  -operating_system="Linux"
emcli list_guest_vm  -server_pool_name="Oracle Server Pool"
-server_name="st-vs1.oracle.com" -operating_system="Linux"
```

list_masking_definitions

Gets the list of masking definitions for an associated target and its script status.

Format

```
emcli list_masking_definitions
     [-definition_name=masking_defn_name_filter]
     [-target_name=target_name_filter]
     [-string_match]
     [-script | -format=[name:<pretty|script|csv>];
                        [column_separator:"column_sep_string"];
                        [row_separator:"row_sep_string"];
     ]
     [-noheader]


[ ] denotes that the parameter is optional
```

Options

- **definition_name**

  Masking definition name. Can be either a full value or a pattern match (%).

- **target_name**

  Database target name. Can be either a full value or a pattern match (%).

- **string_match**

  Uses an exact string match for a target_name and definition_name match.

- **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.
  - `format=`column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

- **noheader**

  Suppresses printing of column headers.

Output Columns

Masking Definition, Database, Status

Examples

Example 1

The following example lists all masking definitions:

```
emcli list_masking_definitions
```

Example 2

The following example lists the masking definition named mask_hr_data :

```
emcli list_masking_definitions -definition_name=mask_hr_data
```

Example 3

The following example lists all masking definitions with names starting with credit_ card :

```
emcli list_masking_definitions  -definition_name=credit_card%
```

Example 4

The following example lists all masking definitions created on a database named testdb :

```
emcli list_masking_definitions -target_name=testdb
```

Example 5

The following example lists all masking definitions created on databases with names starting with test :

```
emcli list_masking_definitions -target_name=test%
```

Example 6

The following example lists the masking definition named mask_hr_data created on a database named testdb :

```
emcli list_masking_definitions -definition_name=mask_hr_data -target_name=testdb
```

Example 7

The following example lists all masking definitions with names starting with credit and created on databases with names starting with test :

```
emcli list_masking_definitions -definition_name=credit% -target_name=test%
```

Example 8

The following example lists all masking definitions without printing the column headers:

```
emcli list_masking_definitions -noheader
```

list_ovm_virtual_server_pool

Lists virtual server pools that the OVM Manager is managing. Note that the OVM Manager's repository should be accessible to OMS.

Note: Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:

https://support.oracle.com

Note: When you execute this EMCLI verb, you are prompted to enter the following value in non-echo mode (if not already provided):

ovm_rep_password

Format

```
emcli list_ovm_virtual_server_pool
          -ovm_rep_desc="OVM Manager repository connect string"
          -ovm_rep_username="OVM Manager repository username"
          -ovm_rep_password="OVM Manager repository password"
          [-ovm_sp_name="OVM Virtual Server Pool name"]
```

Options

- **ovm_rep_desc**

  Connect string of the OVM Manager repository.

  Format:

  ```
  (DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
  (HOST=myovm.example.com)(PORT=my_ovm_port)))(CONNECT_DATA=
  (SID=my_ovm_service)))
  ```

- **ovm_rep_username**

  Connect user name of the OVM Manager repository.

- **ovm_rep_password**

  Connect password of the OVM Manager repository.

- **ovm_sp_name**

  Name of the Virtual Server Pool. If not specified, all accessible Virtual Server Pools are listed.

Output

Virtual Server Pool details, including the Virtual Server Pool name and its associated Virtual Server(s), name(s), and IP(s).

Sample output:

```
Virtual Server Pool Name: vsp0
Master Virtual Server Name: vsp0_master
Master Virtual Server IP: 1.1.1.1
Virtual Server Name: vsp0_vs1
```

```
Virtual Server IP: 1.1.1.2
```

Example

The following example lists the Virtual Server Pool details of my_ovs_sp1., which is available in the remote OVM Manager's repository specified by the connect string.

```
emcli list_ovm_virtual_server_pool
  -ovm_rep_desc="(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
    (HOST=myovm.example.com)(PORT=1521)))(CONNECT_DATA=(SID=xe)))"
  -ovm_rep_username= "my_rep_username"
  -ovm_rep_password= "my_rep_password"
  -ovm_sp_name= "my_ovs_sp1"
```

list_privilege_delegation_settings

Lists privilege delegation setting templates available on the server that apply to targets.

Format

```
emcli list_privilege_delegation_settings
       [-setting_type="SUDO/POWERBROKER]"
       [-noheader]
       [-script | -format=
                [name:<pretty|script|csv>];
                [column_separator:"column_sep_string"];
                [row_separator:"row_sep_string"];
       ]

[ ] denotes that the parameter is optional
```

Options

- **setting_type**

  Setting type. All applicable settings are displayed if you do not specify this option.

- **noheader**

  Displays 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.

  – format=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

  – row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Examples

```
emcli list_privilege_delegation_settings
  -setting_type="SUDO"
```

list_target_privilege_delegation_settings

Lists current privilege delegation settings for targets.

Format

```
emcli list_target_privilege_delegation_settings
      -target_names="name1;name2;name3"
      [-input_file="FILE:file_path"]
      [-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_names**

  List of targets. All targets must be of the host type. Either target_names or input_file must be present.

- **input_file**

  Path of the file that has the list of targets. The file should have one target name per line.

- **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.
  - `format`=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Examples

```
emcli list_target_privilege_delegation_settings
       -target_names="host.oracle.com;host2.oracle.com;

emcli list_target_privilege_delegation_settings
```

```
          -input_file="FILE:/home/nqureshi/targets.txt"

emcli list_target_privilege_delegation_settings
          -target_names="host.oracle.com;host2.oracle.com;
```

list_virtual_server

Lists all virtual servers.

> **Note:** Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:
>
> https://support.oracle.com

Format

```
emcli list_virtual_server
      [-server_pool_name=<Server Pool Name>]

[ ] denotes that the parameter is optional
```

Options

- **server_pool_name**

  Name of the server pool. Lists all virtual servers in the specified virtual server pool.

Output Columns

Name, Server type, Status, Server pool name, Is master, Monitoring server name

Examples

```
emcli list_virtual_server  -server_pool_name="Oracle Server Pool"
```

list_virtual_server_pool

Lists all virtual server pools.

> **Note:** Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:
>
> https://support.oracle.com

Format

```
emcli list_virtual_server_pool
```

Output Columns

Name, Type, Is HA enabled, Master virtual server

Examples

```
emcli list_virtual_server_pool
```

loader_perf

Executes a performance test to determine the network bottleneck between OMS and the Enterprise Manager repository. Only a SYSMAN user can execute this verb.

Format

```
emcli loader_perf
[-batchSize="batch size 1" -batchSize="batch size 3" -batchSize="batch size 3"
   ...]
[-commitSize="commit size 1" -commitSize="commit size 2" -commitSize="commit size
   3" ...]
[-dataSize="data size"]

[ ] denotes that the parameter is optional
```

Options

- **batchSize**

  Batch size for the performance test. Multiple values are allowed. The Default values are {14,50,1}.

- **commitSize**

  Batch size for the performance test. Multiple values are allowed. Default values are {14,50,1}.

- **dataSize**

  Number of records to be inserted for a test, which should be greater than a multiple of 100. If not a multiple of 100, a multiple of 100 less than the given value is considered the dataSize value. The default value is 10000.

Examples

Example 1

The following example displays the time taken to load 10000 records for the default values of batchSize and commitSize.

```
emcli loader_perf
```

Example 2

The following example displays the time taken to load 100000 records for batchSize as {15,40} and commitSize as {10,100}..

```
emcli loader_perf -b=15 -b=40 -c=10 -c=100 -d=100000
```

login

Establishes a new session between the OMS and the EM CLI client. This verb and the Logout verb are useful when you need to run a particular verb as a different user.

The login verb requires Enterprise Manager credentials (or Single Sign-On credentials or both) to log in and establish a session. You do not need to provide the same credentials you provided during setup. The credentials you provide during emcli login overwrite the setup credentials.

Always provide the exact credentials. Providing extra credentials may cause erroneous behavior that you can usually resolve by running emcli setup.

If Enterprise Manager is not SSO-enabled, you should only provide Enterprise Manager credentials using the -username and -password options. For an SSO-enabled Enterprise Manager, if the SSO user is also the Enterprise Manager user, you should only provide SSO credentials with the -ssousername and -ssopassword options. Otherwise, you must provide both Enterprise Manager credentials and SSO credentials.

Note: To avoid an uncommon occurrence in which multiple emcli sessions are created on the OMS, Oracle recommends that you enter the login command before running a script containing EMCLI commands.

Tip: See also logout on page 2-175.

Format

```
emcli login
      -username=<EM Console Username>
      [-password=<EM Console Password>]
      [-ssousername=<EM SSO Username>]
      [-ssopassword=<EM SSO Password>]
      [-force]
```

- Non SSO-enabled Enterprise Manager:

  ```
  emcli login -user=<user_name> -pass=<user_password>
  ```

- SSO-enabled Enterprise Manager — SSO user is also Enterprise Manager user:

  ```
  emcli login -ssouser=<sso_user_name> -ssopass=<sso_user_password>
  ```

- SSO-enabled Enterprise Manager — SSO user is not the Enterprise Manager user:

  ```
  emcli login -user=<user_name> -ssouser=<sso_user_name> -pass=<user_password>
  -ssopass=<sso_user_password>
  ```

Options

- **username**

  Enterprise Manager user name to be used by all subsequent emcli commands when contacting the OMS.

- **password**

  Enterprise Manager user password. If you do not specify this option, you are prompted for the password interactively.

  Note: Providing a password on the command line is insecure and should be avoided.

- **ssousername**

  Enterprise Manager SSO user name to be used by all subsequent emcli commands when contacting the OMS. Provide this value only if Enterprise Manager is configured to use SSO.

- **ssopassword**

  Enterprise Manager SSO password. If you do not specify this option, you are prompted for the password interactively.

  Note: Providing a password on the command line is insecure and should be avoided.

- **force**

  Forces a login even if there is an existing session.

Examples

The following example shows a login as a different user for a non SSO-enabled Enterprise Manager using newly specified credentials, then a subsequent login using the previous credentials.

```
emcli logut
emcli login -user=new_user -pass=new_user_pass
emcli <verb-name>
emcli logout
emcli login -user=old_user -pass=old_user_pass
```

logout

Terminates the existing session with the OMS. This verb and the Login verb are useful when you need to run a particular verb as a different user. After a logout, you need to invoke either the Setup verb or Login verb before invoking any other emcli verb.

Tip: See also login on page 2-173.

Format

```
emcli logout
```

Options

None.

Examples

The following example shows a login as a different user for a non SSO-enabled Enterprise Manager using newly specified credentials, then a subsequent login using the previous credentials.

```
emcli logut
emcli login -user=new_user -pass=new_user_pass
emcli <verb-name>
emcli logout
emcli login -user=old_user -pass=old_user_pass
```

migrate_vsp_ovm_to_em

Provides an automated migration solution that seamlessly migrates the Virtual Server Pool (including Virtual Servers, VMs, and other resources belonging to the server pool) managed by the OVM Manager into Enterprise Manager. This verb does not disrupt the virtualized environment, meaning that no downtime will occur for all GVMs.

Prerequistes

Before starting the migration, ensure that the following prerequisites are met:

- Before you can use this verb or any other virtualization verb, you need to install a one-off patch. Refer to My Oracle Support note 781879.1 for information:

  https://support.oracle.com

- It is advisable to back up the data from the OVM Manager before starting the migration process. The *OVM Manager User's Guide* provides instructions on this process.
- The iptables/ovs Agents of the virtual server might be configured to allow connections with only a specific set of servers. Before starting the migration, ensure that the Enterprise Manager OMS and Monitoring Agent can access (including the SSH connection) the virtual server.
- Executing the topological changing operations (on the server pool being migrated) using OVM Manager during migration adversely affects the process. It is advisable to shut down the OVM manager (console portion only) before starting the migration. However, the OVM database should be up.
- The OVM Manager's repository should be accessible to the OMS.
- The post migration OVM Manager should not be used.

Format

```
emcli migrate_vsp_ovm_to_em
  -ovm_rep_desc="OVM Manager repository connect string"
  -ovm_rep_username="OVM Manager repository username"
  -ovm_rep_password="OVM Manager repository password"
  -ovm_sp_name="OVM Virtual Server Pool name"
  [-ms_details="Monitoring Server details list"]
  [-vs_details="Virtual Server details list"]
  [-input_file="FILE:filename"]
  [-new_em_sp_name="New Virtual Server Pool name"]
  [-on_error="rollback|stop"]
  [-repair="Job ID"]
  [-clean]
  [-rollback="Job ID"]
```

Note: When you execute this EMCLI verb, you are prompted to enter the following values in non-echo mode if they are required but not specified:

- OVM repository password
- Virtual Server host password
- Monitoring Server password

Options

- **ovm_rep_desc**

  Connect string of the OVM Manager repository.

  Format:

  ```
  (DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
  (HOST=myovm.example.com)(PORT=my_ovm_port)))(CONNECT_DATA=
  (SID=my_ovm_service)))
  ```

- **ovm_rep_username**

  Connect user name of the OVM Manager repository.

- **ovm_rep_password**

  Connect password of the OVM Manager repository.

- **ovm_sp_name**

  Name of the Virtual Server Pool to be migrated from OVM Manager to Enterprise Manager.

- **ms_details**

  List of Monitoring Server details. Use the following format to specify:

  ```
  host1:port1:username1:password1;host2:port2:username2:password2...
  ```

- **vs_details**

  List of Virtual Server details in the Virtual Server Pool to be migrated. Use the following format to specify:

  ```
  vs_host1:username1:password1:monitoring_server_host1:monitoring_server_
  port1:proxy_home1;...
  ```

- **input_file**

  File describing Monitoring Server details and Virtual Server details.

  Example:

  ```
  monitoring_server_details="host1:port1:username1:password1"
  monitoring_server_details="host2:port2:username2:password2"
  ...
  ...
  virtual_server_details="host1:username1:password1:monitoring_server_
  host1:monitoring_server_port1:proxy_home1"
  virtual_server_details="host2:username2:password2:monitoring_server_
  host2:monitoring_server_port1:proxy_home2"
  ...
  ...
  ```

  Note: if you specify -input_file, -ms_details and -vs_details are ignored.

- **new_em_sp_name**

  New name of the Virtual Server Pool to be migrated. It should not clash with the existing virtual server pool available in Enterprise Manager. If not specified, the value specified in -ovm_sp_name is used.

- **on_error**

Defines the behavior of the migration operation if an error occurs. The rollback value removes the partial data in Enterprise Manager if an error occurs. The stop value does not remove the partial data if an error occurs so that you can execute this verb again with the repair option (and providing the Job ID).

- **repair**

  Job ID to perform repairing migration. By specifying -repair, you only need to provide amended options. Migration then continues from the last succeeded step.

  Note: Only stop mode running is repairable by future runs.

- **clean**

  Set this flag to clean the specified OVM Virtual Server Pool data after a successful migration.

- **rollback**

  Job ID to roll back. This operation forces clean the partial data the migration job creates in Enterprise Manager, including the Virtual Server Pool and its associated Virtual Server targets.

Output

Job name, Job ID, and Job Execution ID.

Note that future executions of the same migration task (such as repair and rollback) require the same Job ID.

Sample output:

```
Job details:
Job name = OVMEMMigrationJob_2010-03-11_15-09-44
Job ID = 818E82AE0318AD90E040449820C47F2D
Job execution ID = 818E82AE031AAD90E040449820C47F2D
```

Examples

Example 1

The following example shows migration with rollback as an on_error option.

This operation migrates the Virtual Server Pool my_ovs from OVM Manager to Enterprise Manager and changes the name to my_new_ovs_sp1. my_ovs contains two Virtual Servers: vs1.example.com and vs2.example.com. After migration, Monitoring Servers ms1.example.com:3872 and ms2.example.com:1830 respectively will monitor the virtual servers. If the job fails, no partial data will be left in the Enterprise Manager repository, since the on_error option is rollback.

This verb functions only on versions 2.1.2, 2.1.5, and 2.2 of the OVM Manager.

```
emcli migrate_vsp_ovm_to_em
  -ovm_rep_desc= "(DESCRIPTION=(ADDRESS_
LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=myovm.example.com)(PORT=1521)))(CONNECT_
DATA=(SID=xe)))"
  -ovm_rep_username= "my_rep_username"
  -ovm_rep_password= "my_rep_password"
  -ovm_sp_name= "my_ovs_sp1"
  -new_em_sp_name= "my_new_ovs_sp1"
  -ms_details=
   "ms1.example.com:3872:my_ms_username1:my_ms_password1;
    ms2.example.com:1830:my_ms_username2:my_ms_password2"
  -vs_details=
```

```
   "vs1.example.com:vs1_root:vs1_root_password:ms1.example.com:3872:/tmp;
    vs2.example.com:vs2_root:vs2_root_password:ms2.example.com:1830:/tmp"
  -on_error= "rollback"
```

Example 2

The following example shows migration with stop as an on_error option. The operation is the same as for Example 1, except that new data loaded in the Enterprise Manager repository is not cleaned up when the job fails. The most common reasons for the job failure are typos while providing details and incorrect passwords.

```
emcli migrate_vsp_ovm_to_em
  -ovm_rep_desc= "(DESCRIPTION=(ADDRESS_
LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=myovm.example.com)(PORT=1521)))(CONNECT_
DATA=(SID=xe)))"
  -ovm_rep_username= "my_rep_username"
  -ovm_rep_password= "my_rep_password"
  -ovm_sp_name= "my_ovs_sp1"
  -new_em_sp_name= "my_new_ovs_sp1"
  -ms_details=
   "ms1.example.com:3872:my_ms_username1:my_ms_password1;
    ms2.example.com:1830:my_ms_username2:my_ms_password2"
  -vs_details=
   "vs1.example.com:vs1_root:vs1_root_password:ms1.example.com:3872:/tmp;
    vs2.example.com:vs2_root:vs2_root_password:ms2.example.com:1830:/tmp"
  -on_error= "stop"
```

Example 3

The following example repairs a failed migration job. Suppose that the job submitted (assume the Job ID is 7BFE0457A7E39410E040449820C4093E) in example 2 failed because of an incorrect Monitoring Server Password (my_fixed_ms_password1 is the correct one). You can supply the correct password (ms_details option) as shown below and also provide the repair option with the Job ID of the previous submission so that the job can re-run from the step where it failed.

```
emcli migrate_vsp_ovm_to_em
  -ms_details= "ms1.example.com:3872:my_ms_username1:
 my_fixed_ms_password1;ms2.example.com:1830:
 my_ms_username2:my_ms_password2"
  -repair="7BFE0457A7E39410E040449820C4093E"
```

Example 4

The following example of migration with the -input_file option functions like Example 1, except that it reads Monitoring Sever and Virtual Server details from /path/to/migration.txt.

```
emcli migrate_vsp_ovm_to_em
  -ovm_rep_desc="(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
     (HOST=myovm.example.com)(PORT=1521)))(CONNECT_DATA=(SID=xe)))"
  -ovm_rep_username="my_rep_username"
  -ovm_rep_password="my_rep_password"
  -INPUF_FILE= "FILE:/path/to/migration.txt"
  -on_error="stop"
```

File content of /path/to/migration.txt:

```
monitoring_server_details=" ms1.example.com:3872:
my_ms_username1:my_ms_password1"
monitoring_server_details="ms2.example.com:1830:
my_ms_username2:my_ms_password2"
```

```
virtual_server_details="vs1.example.com:vs1_root:
vs1_root_password:ms1.example.com:3872:/tmp"
virtual_server_details="vs2.example.com:vs2_root:
vs2_root_password:ms2.example.com:1830:/tmp"
```

Example 5

The following example rolls back the partial data (including targets created) migrated in Enterprise Manager by the migration job with Job ID (of example2) 7BFE0457A7E39410E040449820C4093E .

```
emcli migrate_vsp_ovm_to_em
          -rollback=7BFE0457A7E39410E040449820C4093E
```

modify_aggregate_service

Modifies an aggregate service instance.

Format

```
emcli 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"]
      [-privilege_propagation=true/false]
      [-drop_existing_grants=yes/no]

[ ] 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.

- **privilege_propagation**

  Enables/disables the privilege propagation flag for the group. Converts the normal group to a privilege propagating group and vice versa. Possible values are true or false.

- **drop_existing_grants**

  Drops existing grants on a privilege propagating group that was converted with the privilege_propagation option. This option is only valid with the privilege_propagation option. Possible values are yes or no. The default is yes.

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_collection_schedule

Modifies the collection schedule of a collection setup for metrics and policies for the specified set of targets. Combining all the metrics, running a script, and collecting the data is referred to as a collection. The collection has various attributes associated with it, such as the collection schedule, upload frequency, and so forth.

Format

```
emcli modify_collection_schedule
      -targetType=ttype
      -targetNames=tname1;tname2;tname3...
      -collectionName=collname
      [-collectionStatus=Enabled or Disabled]
      [-freqType={Minute}{Hour}{Day}{Week}{Weekly}{Month}
      [-freqValue={any integer value for Minute/Hour/Day/Week}{One or more from
       Mon...Sun for Weekly}{One or more from 1;2..31 or Last for Month}
      [-preview=Y or N]

[ ] denotes that the parameter is optional
{ } denotes that you can select one of the options in the series shown

Note: All of the parameters and choices are case-insensitive
```

Options

- **targetType**

  You must specify a single target type value, and it should be the same as specified in the repository.

  > **Note:** Only individual target types are currently supported.

- **targetNames**

  The target name should be the same as exists in the repository. All of the targets should be the same target type you specified in the targetType parameter. Use a semicolon (;) to separate the names. Changes to the collection schedule will be executed for only valid target name and target type combinations. For example:

  host1;host2;host3

- **collectionName**

  The collection name should be exactly the same as exists in the repository or the corresponding collections .xml file present on the Agent.

  Access files from the following locations to determine the collection to be modified. Select the desired collection and provide it as input to the emcli utility.

  – $AGENT_HOME/sysman/admin/metadata/<targetType>.xml

    This file is shipped as a part of the setup and contains information regarding the metrics for this target type.

  – $AGENT_HOME/sysman/admin/default_collection/<targetType>.xml

    This file is shipped as a part of the setup and contains the collections shipped by default.

  - $AGENT_HOME/sysman/emd/collection/<targetType_targetName>.xml

    Whenever changes have occurred for any particular target, this file is automatically generated. Collections for user-defined metrics are available in this file.

- **collectionStatus**

  Enables or disables the collection. The default is Enabled. If Disabled, freqType and freqValue are ignored.

- **freqType**

  You can specify one of the following values:

  Minute (default)
  Hour
  Day
  Week
  Weekly
  Month

  For Week, you must specify an integer value as the frequency value. For instance, if you specify freqType='WEEK' and freqValue='2', the collection occurs every two weeks.

  For Weekly, the possible values are Mon, Tue, Wed, Thu, Fri, Sat, Sun. For instance, if you specify freqType='Weekly' and freqValue='Tue;Thu;Sun', the collection occurs every Tuesday, Thursday and Sunday of a week.

  The schedule is modified based on your selection. You do not need to specify a value (and the value will be ignored) if the collectionStatus parameter is set to Disabled.

  If you use this parameter, you must also use the freqValue parameter.

- **freqValue**

  You can specify one of the following values:

  - You must specify an integer value if the freqType is any one of Minute, Hour, Day, or Week. The default value is 5.
  - For Weekly, specify one or more choices from Mon, Tue, Wed, Thu, Fri, Sat, and Sun. If the collection occurs on any particular day(s) of the week, you must specify the corresponding value(s) against the Weekly option.
  - For Monthly, specify one or more choices from 1...31 or Last. If the collection occurs on any particular date(s) in a month, you must specify the corresponding value(s) against the Monthly option.

  You do not need to specify a value (and the value will be ignored) if the collectionStatus parameter is set to Disabled.

  If you use this parameter, you must also use the freqType parameter.

- **preview**

  Provides a preview of the changes that would occur if this verb is executed. The default value for this option is Y (Yes), whether you specify the option or not. If you specify N, the changes to the collection schedule are executed for both the repository and Agent.

Examples

Example 1

The following example changes the collection schedule to collect once every 5 minutes for hosts host1, host2, and host3. DiskActivity is a collection item associated with a host target type. The preview flag is set to Y, so the changes are not executed, but you can see the metrics affected if the changes were implemented.

```
emcli modify_collection_schedule -targetType="host"
      -targetNames="host1;host2;host3" -collectionName="DiskActivity"
      -freqType="Minute" -freqValue="5" -preview="Y"
```

Example 2

The following example changes the collection schedule to collect once every 15 hours for host host1. Inventory is a collection item associated with a host target type. The preview flag is set to N, so the changes are executed for the associated metrics for both the repository and Agent.

```
emcli modify_collection_schedule -targetType="host"
      -targetNames="host1" -collectionName="Inventory"
      -freqType="Hour" -freqValue="15" -preview="N"
```

Example 3

The following example changes the collection schedule to collect on Monday and Thursday every week for hosts host1 and host2. Inventory is a collection item associated with a host target type. The preview option is not specified, but since the value is Y whether you specify the option or not, the changes are not executed, but you can see the metrics affected if the changes were implemented.

```
emcli modify_collection_schedule -targetType="host"
      -targetNames="host1;host2" -collectionName="Inventory"
      -freqType="Weekly" -freqValue="Mon;Thu"
```

Example 4

The following example changes the collection schedule to collect on the 1st, 5th, 23rd, and last day of every month for hosts host1 and host2. Inventory is a collection item associated with a host target type.

```
emcli modify_collection_schedule -targetType="host"
      -targetNames="host1;host2" -collectionName="Inventory"
      -freqType="Month" -freqValue="1;5;23;Last"
```

Example 5

The following example disables the collection schedule for hosts host1 and host2. Inventory is a collection item associated with a host target type.

```
emcli modify_collection_schedule -targetType="host"
      -targetNames="host1;host2" -collectionName="Inventory"
      -collectionStatus="Disabled"
```

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

```
emcli modify_group
      -name="name"
      [-type=<group>]
      [-add_targets="name1:type1;name2:type2;..."]...
      [-delete_targets="name1:type1;name2:type2;..."]...
      [-privilege_propagation=true/false]
      [-drop_existing_grants=yes/no]

[ ] 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`. You can specify this option more than once.

- **delete_targets**

  Targets to delete, each specified as `target_name:target_type`. You can specify this option more than once.

- **privilege_propagation**

  Enables/disables the privilege propagation flag for the group. Converts the normal group to a privilege propagating group and vice versa. Possible values are true or false.

- **drop_existing_grants**

  Drops existing grants on a privilege propagating group that was converted with the privilege_propagation option. This option is only valid with the privilege_propagation option. Possible values are yes or no. The default is yes.

Examples

Example 1

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
```

Example 2

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

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

Example 3

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
```

Example 4

The following example converts group my_group to privilege propagation, ignores if already converted, and drops all of its existing grants.

```
emcli modify_group -name=my_group
      -privilege_propagation=true
```

Example 5

The following example converts group my_group to non-privilege propagation, ignores if already converted, and retains all of its existing grants on my_group.

```
emcli modify_group -name=my_group
      -privilege_propagation=false
      -drop_existing_grants=no
```

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

```
emcli 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`. You can specify this option more than once.

- **delete_targets**

  Targets to delete, each specified as `target_name:target_type`. You can specify this option 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_redundancy_group

Modifies a redundancy group.

Format

```
emcli modify_redundancy_group
        -redundancyGroupName="redGrpName"
        [-owner="new owner"]
        [-memberTargetType="tType"]
        [-add_targets="tName1;tName2"]
        [-delete_targets="tName3;tName4"]
        [-group_status_criterion="NUMBER" or "PERCENTAGE"]
        [-group_status_tracked="UP" or "DOWN"]
        [-group_status_value=(see the Options section)]
        [-privilege_propagation=true/false]
        [-drop_existing_grants=yes/no]

[ ] denotes that the parameter is optional
```

Options

- **redundancyGroupName**

  Name of the redundancy group.

- **owner**

  Valid owner to be specified.

- **memberTargetType**

  Target type of the constituent member targets. You need to specify this parameter if you specify either add_targets or delete_targets.

- **add_targets**

  Member targets to be added to this redundancy group.

- **delete_targets**

  Member targets to be deleted from this redundancy group.

- **group_status_criterion**

  This option and the next two calculate the status of the Redundancy Group. Consequently, you need to specify all three options together. If this is not to be a capacity group, you need to specify the following combination:

  ```
  -group_status_criterion='NUMBER' -group_status_tracked='UP' -group_status_
  value='1']
  ```

- **group_status_tracked**

  See the option above.

- **group_status_value**

  See the group_status_criterion option.

  You can specify any value between 1 and 100 if -group_status_criterion= "PERCENTAGE", or any value between 1 and the number of targets present if -group_status_criterion="NUMBER".

- **timezone_region**

  Time zone region of this redundancy group. For a list of valid time zone regions, enter the following command at SQLPLUS:

  ```
  SELECT TZNAME FROM V$TIMEZONE_NAMES
  ```

  You may need to have the SELECT_CATALOG_ROLE role to execute this command.

- **privilege_propagation**

  Enables/disables the privilege propagation flag for the group. Converts the normal group to a privilege propagating group and vice versa. Possible values are true or false.

- **drop_existing_grants**

  Drops existing grants on a privilege propagating group that was converted with the privilege_propagation option. This option is only valid with the privilege_propagation option. Possible values are yes or no. The default is yes.

Examples

The following example changes the configuration of the 'redGrp1' redundancy group to add listener, listener2, and listener3 to its existing members, and delete listener4 and listener5 from its existing members.

```
emcli modify_redundancy_group -redundancyGroupName='redGrp1'
       -memberTargetType='oracle_listener'
       -add_targets='listener;listener2;listener3'
       -delete_targets='listener4;listener5'
       -group_status_criterion='NUMBER'
       -group_status_tracked='UP'
       -group_status_value='2'
```

modify_role

Modifies an existing Enterprise Manager administrator role.

> **Note:** Omit an argument to leave its value unchanged.
>
> To update a role and add targets to the role, use the grant_privs verb.

Format

```
emcli 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**

  Replaces the description of the role.

- **roles**

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

- **users**

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

- **privilege**

  Replaces privileges granted to this role. You can specify this option 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

Example 1

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.example.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.example.com:host"
      -users="johndoe;janedoe"
```

Example 2

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

```
emcli modify_system
      -name="name"
      [-type=<generic_system>]
      [-add_members="name1:type1;name2:type2;..."]...
      [-delete_members="name1:type1;name2:type2;..."]...
      [-owner="new_owner"]
      [-privilege_propagation=true/false]
      [-drop_existing_grants=yes/no]

[ ] 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 this option more than once.

- **delete_members**

  Targets to delete, each specified as `target_name:target_type`. You can specify this option more than once.

- **owner**

  New owner of the system.

- **privilege_propagation**

  Enables/disables the privilege propagation flag for the group. Converts the normal group to a privilege propagating group and vice versa. Possible values are true or false.

- **drop_existing_grants**

  Drops existing grants on a privilege propagating group that was converted with the privilege_propagation option. This option is only valid with the privilege_propagation option. Possible values are yes or no. The default is yes.

Examples

Example 1

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
```

Example 2

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

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

Example 3

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

```
emcli 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**

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

- **subseparator=properties**

  Specifies 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**

  Sets the target display name.

- **on_agent**

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

Examples

Example 1

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"
```

Example 2

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
```

Example 3

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.example.com"
      -separator=properties="|"
      -subseparator=properties="="
      -on_agent
```

Example 4

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

```
emcli modify_user
      -name="name"
      [-password="password"]
      [-roles="role1;role2;..."]
      [-email="email1;email2;..."]
      [-privilege="name;[[target_name:target_type]|jobid]"]...
      [-profile="profile_name"]
      [-desc="user_description"]
      [-expired="true/false"]
      [-prevent_change_password="true/false"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Administrator name.

- **password**

  Replaces the administrator password with the specified password.

- **roles**

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

- **email**

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

- **privilege**

  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

- **profile**

  Database profile name. When not passed, this value is not altered.

- **desc**

  User description

- **expired**

  True immediately expires the password. The default is false.

- **prevent_change_password**

  True prevents a user from updating his/her password. The default is false.

Examples

Example 1

The following example modifies the new_admin administrator. The user will have two privileges: to view the job with ID 923470234ABCDFE230184947530911111 and to view the target host1.example.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;923470234ABCDFE230184947530911111"
      -privilege="view_target;host1.example.com:host"
```

Example 2

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=""
```

pause_guest_vm

Pauses a guest Virtual Machine. To pause the guest Virtual Machine, it should be in the Running state.

Tip: See also unpause_guest_vm on page 2-283.

Format

```
emcli pause_guest_vm
      -guest_vm_name=<Virtual Machine Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- **server_pool**

  Name of the guest server pool.

Examples

The following example pauses the dom15 guest Virtual Machine.

```
emcli pause_guest_vm -guest_vm_name="dom15" -server_pool_name="Oracle Server Pool"
```

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

```
emcli 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 the 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.example.com"
      -stageserver="stageserver.example.com:/private/share"
      -stgcredentials="joe"
      -schedule="type:immediate"
      -resettimeout="100"
      -target="mylabel"
      -input_file="config_properties:properties.txt"
      -assignment="provision mylabel"
```

reassoc_masking_definition

Reassociates an existing masking definition with another database target.

Format

```
emcli reassoc_masking_definition
   -definition_name=masking definition name
   -target_name=database target name
   [-parameters=name1:value1;name2:value2;...]
   [-credential_set_name=credential_set_name]
   [-input_file=parameter_tag:file_path]

[ ] denotes that the parameter is optional
```

Options

- **definition_name**

  Masking definition name.

- **target_name**

  New database target name to associate the masking definition with.

- **parameters**

  List of name-value pairs that represent the credentials required for connecting to the database instance. The supported parameters are db_username, db_password, and db_role.

- **credential_set_name**

  Set name of the preferred credentials stored in the Enterprise Manager repository. The valid values for this parameter are:

  – **DBCredsNormal** —

    Default normal credential set for an oracle_database target. If no value is specified, DBCredsNormal is used.

  – **DBCredsSYSDBA** —

    SYSDBA credential set for an oracle_database target.

  You can only specify this parameter when the override credential parameters such as db_username and db_password are not present.

- **input_file**

  Used in conjunction with the parameters option, this option enables you to store parameter values, such as username and password, in a separate file. This option specifies a mapping between a tag and a local file path. The tag is specified in lieu of specific parameter values for the parameters option. The tag must not contain colons (:) or semi-colons (;).

Output

Success or failure message along with the details.

Examples

Example 1

The following example reassociates the masking definition mask_hr_data with the new database target testdb2 :

```
emcli reassoc_masking_definition
     -definition_name=mask_hr_data
     -target_name=testdb2
     -parameters="db_username:system;db_password:password;db_role:NORMAL"
```

Example 2

The following example reassociates the masking definition mask_hr_data with the new database target testdb2. The database password is read from the pwd.txt file.

```
emcli reassoc_masking_definition
      -definition_name=mask_hr_data
      -target_name=testdb2
      -parameters="db_username:system;db_password:PWD_FILE;db_role=SYSDBA"
      -input_file="PWD_FILE:pwd.txt"
```

Example 3

The following example reads the credentials from the preferred credential set DBCredsNormal and reassociates the masking definition.

```
emcli reassoc_masking_definition
-definition_name=mask_hr_data
-target_name=testdb2
```

Example 4

The following example reads the credentials from the preferred credential set DBCredsSYSDBA and reassociates the masking definition.

```
emcli reassoc_masking_definition
-definition_name=mask_hr_data
-target_name=testdb2
-credential_set_name=DBCredsSYSDBA
```

reboot_guest_vm

Reboots a guest virtual machine. To reboot the guest virtual machine, it should be in the Running state.

Format

```
emcli reboot_guest_vm
      -guest_vm_name=<Virtual Machine Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine name.

- **server_pool_name**

  Name of the server pool.

Examples

The following example reboots the dom15 guest Virtual Machine.

```
emcli reboot_guest_vm -guest_vm_name="dom15" -server_pool_name="Oracle Server
Pool"
```

reboot_virtual_server

Reboots a virtual server. To reboot the virtual server, it should be in the Up state.

Format

```
emcli reboot_virtual_server
      -server_name=Server name
```

Options

- **server_name**

  Name of the virtual server.

Examples

The following example reboots the st-vs1.example.com virtual server.

```
emcli reboot_virtual_server -server_name="st-vs1.example.com"
```

relocate_targets

Format

```
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 target to be relocated>
       -changed_param=<propName>:<propValue>
       -input_file=dupTargets:<targets contents>
       -input_file=moveTargets:"complete path to file containing targets with
         overridden property values"
       -copy_from_src [-changed_param=<propName>:<propValue>]*
       [-ignoreTimeSkew=yes]
       [-force=yes]

[ ] denotes that the parameter is optional
```

> **Note:** To relocate a composite target, you must specify the input_file=dupTargets option, and you must not combine -target_type or -target_name.

Modes

There are two modes for this verb:

- **Create Mode**

  This mode 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>;
        [-ignoreTimeSkew=yes]
  ```

  > **Tip:** See the Examples section for more samples of the create mode.

- **Exist Mode**

  In this mode, the target also exists at the destination.

  ```
  emcli relocate_targets
        -src_agent=<source agent target name>
        -dest_agent=<destination agent target name>
        -target_name=<target name>
        -target_type=<target type>
        [-ignoreTimeSkew=yes]
        [-force=yes]
  ```

In all cases, relocation moves all collections and blackouts for these targets from the source Agent to destination Agent, and makes the destination Agent the monitoring Agent for these targets in Enterprise Manager.

Options

- **src_agent**

  Agent currently monitoring the targets. If srcAgent is not known, enter currentOwner as the argument.

- **dest_agent**

  Agent that should monitor the targets.

- **target_name**

  Name of the target that needs to be moved.

- **target_type**

  Type of target that needs to be moved.

- **changed_param**

  The value of the propName property in the target should be changed to propValue.

- **input_file=dupTargets**

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

  To relocate a composite target, you must specify the input_file=dupTargets option, and you must not combine -target_type or -target_name.

- **input_file=moveTargets**

  Takes a file name that contains a list of targets, one per line, in the following format:

  ```
  <targetType>:<targetName>[;<propName>=<propValue>]*
  ```

- **copy_from_src**

  Target properties should be copied from the source Agent

- **ignoreTimeSkew**

  If specified, the target is relocated, ignoring the time skew between the source and destination Agent.

- **force**

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

Examples

Example 1

The following Create Mode example creates a target on the destination Agent by copying the target property content from the source Agent, while allowing some property values to be changed.

```
emcli relocate_targets -src_agent=<source agent>
      -dest_agent=<destination agent>
      -target_name=<target name>
      -target_type=<target type>
```

```
      -copy_from_src
      [-ignoreTimeSkew=yes]
      [-changed_param=<Propname>:<Value>]*
```

Example 2

The following Create Mode example creates a list of targets on the destination Agent specified in the moveTargets file. You can specify property value overrides.

```
emcli relocate_targets -src_agent=<source agent>
      -dest_agent=<destination agent>
      -input_file=moveTargets:<complete file path>
      [-ignoreTimeSkew=yes]
```

remove_beacon

Removes a beacon from the monitoring set of beacons.

Format

```
emcli 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**

  Beacon name to remove.

- **forceRemove**

  If specified, skips the sanity checks for availability definition.

Examples

The following example removes MyBeacon from the 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

```
emcli 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'
```

remove_target_property

Removes the target property from all targets of the specified target type. This also removes all values associated with this target property.

Format

```
emcli remove_target_property
      -target_type="target_type"
      -property="property_name"
```

Options

- **target_type**

  Target type for which you want to remove this property. To remove this property from all target types for which it is defined, you can specify the "*" wildcard character.

- **property**

  Name of the property you want to remove. Property names are case-sensitive. You cannot remove the following Oracle-provided target properties:

  Comment, Deployment Type, Line of Business, Location, Contact

Examples

Example 1

The following example removes the target property Owner from all targets of type oracle_database. This also removes all values associated with this target property.

```
emcli remove_target_property  -target_type="oracle_database" -property="Owner"
```

Example 2

The following example removes the target property Owner from all targets. This also removes all values associated with this property for all target types.

```
emcli remove_target_property  -target_type="*" -property="Owner"
```

reschedule_instance

Reschedules a submitted procedure instance. You can only reschedule scheduled instances.

Format

```
emcli reshedule_instance
      -instance={instance guid}
      -schedule=
          start_time:yyyy/MM/dd HH:mm;
          [tz:{java timezone ID}];
          [grace_period:xxx]
```

Options

- **instance**

  GUID of the instance to execute.

- **schedule**

  Schedule for the procedure instance:

  - **start_time** — When the procedure should start.
  - **tz —** Optional time zone ID.
  - **grace_period** — Optional grace period in minutes.

Examples

```
emcli reschedule_instance -instance=16B15CB29C3F9E6CE040578C96093F61
-schedule="start_time:2006/6/21 21:23;tz:America/New_York;grace_period:60"
```

resume_guest_vm

Resumes a guest Virtual Machine. To resume the guest Virtual Machine, it should be in the Suspended state.

Tip: See also suspend_guest_vm on page 2-279.

Format

```
emcli resume_guest_vm
      -guest_vm_name=<Virtual Machine Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- **server_pool_name**

  Name of the server pool.

Examples

The following example resumes the dom15 guest Virtual Machine.

```
emcli resume_guest_vm -guest_vm_name="dom15" -server_pool_name="Oracle Server
Pool"
```

resume_instance

Resumes a suspended deployment instance.

Format

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

Options

- **instance**

  GUID of the instance.

Examples

```
emcli resume_instance -instance=16B15CB29C3F9E6CE040578C96093F61
```

resync_agent

Performs an Agent recovery. A message is issued if the specified Agent does not exist.

Format

```
emcli resyncAgent
      -agent="Agent Name"
      [-keep_blocked]

[ ] denotes that the parameter is optional
```

Options

- **agent**

  Name of the Agent for which to perform the Agent recovery.

- keep_blocked

  Leaves the Agent blocked even if the resync succeeds. By default, the Agent becomes unblocked if the resync is successful.

Examples

```
emcli resyncAgent -agent="Agent Name"
```

retry_instance

Retries a failed instance or failed step.

Format

```
emcli retry_instance
      -instance=<instance_guid>
      [-stateguid=<state_guid>]

[ ] denotes that the parameter is optional
```

Options

- **instance**

  GUID of the instance.

- **stateguid**

  Comma-separated list of state GUIDs.

Examples

```
emcli retry_instance -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid=51F762417C4943DEE040578C4E087168

emcli retry_instance -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid='51F762417C4943DEE040578C4E087168,51F762417C4944DEE040578C4E087168'
```

retry_job

Restarts a previously failed job execution.

Format

```
emcli 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**

  Displays 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.
  - `format`=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.
  - row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

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_license_no_validation

Revokes licenses on a set of user-specified packs, or all packs to a set of user-specified targets, or all targets belonging to the input licensable target type.

For 11g database targets, you cannot enable or disable the Database Diagnostic and Tuning Packs through the user interface. You need to set the control_management_pack_access initialization parameter to manage your licenses. For information about this parameter, see the Enterprise Database Management chapter of *Oracle Enterprise Manager Licensing Information*.

> **Tip:** You can use this verb to revoke licenses for standalone target types, such as hosts and databases, but you cannot use this verb to revoke licenses for the parent Application Server (oracle_ias) target type, which has dependent target types of OC4J, Jserv, Web Cache, and so forth. To do this, use the revoke_license_with_validation verb instead.
>
> For example, for pack ias_config and an Application Server target of AS1 with an associated dependent target of OC4J1, this verb revokes the license to AS1, but this does not propagate to OC4J1.

Format

```
emcli revoke_license_no_validation
      -type="target_type"
      [-targets="tname1;tname2;..."]
      [-packs="pack1;pack2;..."]
      [-file="file_name"]
      [-displayAllMessages]

[ ] denotes that the parameter is optional
```

Options

- **type**

  Target type as it exists in the database. Names cannot contain colons (:), semi-colons (;), or any leading or trailing blanks. You can specify only one target type at a time; for example, -type="oracle_database".

- **targets**

  Targets should be specified in the following sequence:

  ```
  TargetName1;TargetName2;
  ```

  For example:

  ```
  -targets="database1;database2;database3;"
  ```

  The semi-colon (;) is the target separator.

  See the "Examples" section below for information about providing arguments for the targets option.

- **packs**

  License packs should be specified in the following sequence:

  ```
  pack1;pack2;
  ```

For example:

```
-packs="db_diag;db_config;"
```

The semi-colon (;) is the pack separator.

See the "Examples" section below for information about providing arguments for the packs option.

- **file**

  Specify the file name, including the complete path. For example:

  ```
  -file="/usr/admin1/db_license.txt"
  ```

  The file should contain the list of targets and packs according to the following cases:

  – If you only need to provide a list of targets, use the following format:

    ```
    targets=database1;database2;database3;
    ```

  – If you only need to provide a list of packs, use the following format:

    ```
    packs=db_diag;db_config;
    ```

  – If you need to provide a list of both targets and packs, use the following format:

    ```
    targets=database1;database2;database3;
    packs=db_diag;db_config;
    ```

- **displayAllMessages**

  Displays all messages. Only error messages are displayed by default. "=value" is not allowed on the command line.

Examples

Example 1 and Example 2 below revoke licenses of specific packs for specific targets. In order to know which target types and pack names you can pass as arguments, you can use the view named mgmt_license_view to see a list of licensable targets, their target types, and the list of packs licensed on them.

To obtain this information, do the following:

1. Access SQL*Plus with your username and password, using sysman or other user that has access to sysman.mgmt_license_view.
2. Select a distinct pack name from sysman.mgmt_license_view, where:

   ```
   target_type=<oracle_database>
   ```

The following example shows pack names for an Oracle database you specify as the target type.

```
PACK_NAME
------------------
db_config
provisioning
db_sadm
db_tuning
db_diag
provisioning_db
db_chgmgt
```

```
7 rows selected.
```

Based on this information, to revoke a license to the database1 target for the db_chgmgt pack, you would enter the following command:

```
emcli revoke_license_no_validation -type="oracle_database" -targets="database1"
-packs="db_chgmgt"
```

The only limitation of mgmt_license_view is that it only lists the packs for a target type where the pack is granted to at least one target of that type. That is, if the pack is not granted to any target of that type, mgmt_license_view cannot provide any information.

Example 1

The following example revokes the license of the db_diag and db_config packs to database1, database2, and database3 targets (oracle_database target type):

```
emcli revoke_license_no_validation -type="oracle_database"
          -targets="database1;database2;database3;" -packs="db_diag;db_config;"
```

Example 2

The following example revokes the license of the db_diag and db_config packs to all database targets in the setup:

```
emcli revoke_license_no_validation -type="oracle_database"
       -packs="db_diag;db_config;"
```

Example 3

The following example revokes the license of all packs (applicable to database targets) to database1, database2, and database3 targets in the setup:

```
emcli revoke_license_no_validation -type="oracle_database"
         -targets="database1;database2;database3;"
```

Example 4

The following example revokes the license of all packs (applicable to database targets) to all database targets in the setup:

```
emcli revoke_license_no_validation -type="oracle_database"
```

Example 5

The following example uses a text file to pass targets and pack names as the argument. It revokes the license of the db_diag and db_config packs to the database1, database2, and database3 targets (oracle_database target type):

```
emcli revoke_license_no_validation -type="oracle_database"
          -file="/usr/admin1/db_license.txt"
             targets=database1;database2;database3;
                   packs=db_diag;db_config;
```

where the content of the "/usr/admin1/license/db_license.txt" file is as follows:

```
targets=database1;database2;database3;
packs=db_diag;db_config;
```

revoke_license_with_validation

Revokes licenses on a set of user-specified packs, or all packs to a set of user-specified targets, or all targets belonging to the input licensable target type as per business rules.

For 11g database targets, you cannot enable or disable the Database Diagnostic and Tuning Packs through the user interface. You need to set the control_management_pack_access initialization parameter to manage your licenses. For information about this parameter, see the Enterprise Database Management chapter of *Oracle Enterprise Manager Licensing Information*.

> **Tip:** You can use this verb to revoke licenses for standalone target types, such as hosts and databases, and you also use this verb to revoke licenses for the parent Application Server (oracle_ias) target type, which has dependent target types of OC4J, Jserv, Web Cache, and so forth.
>
> For example, for pack ias_config and an Application Server target of AS1 with an associated dependent target of OC4J1, this verb revokes the license to AS1 and also propagates to OC4J1 (and all other dependent targets associated with AS1).
>
> To revoke licenses for only standalone target types, use the revoke_license_no_validation verb.

Format

```
emcli revoke_license_with_validation
      -type="target_type"
      [-targets="tname1;tname2;..."]
      [-packs="pack1;pack2;..."]
      [-file="file_name"]
      [-displayAllMessages]

[ ] denotes that the parameter is optional
```

Options

- **type**

  Target type as it exists in the database. Names cannot contain colons (:), semi-colons (;), or any leading or trailing blanks. You can specify only one target type at a time; for example, -type="oracle_database".

- **targets**

  Targets should be specified in the following sequence:

  ```
  TargetName1;TargetName2;
  ```

  For example:

  ```
  -targets="database1;database2;database3;"
  ```

  The semi-colon (;) is the target separator.

  See the "Examples" section below for information about providing arguments for the targets option.

- **packs**

  License packs should be specified in the following sequence:

```
pack1;pack2;
```

For example:

```
-packs="db_diag;db_config;"
```

The semi-colon (;) is the pack separator.

See the "Examples" section below for information about providing arguments for the packs option.

- **file**

  Specify the file name, including the complete path. For example:

  ```
  -file="/usr/admin1/db_license.txt"
  ```

  The file should contain the list of targets and packs according to the following cases:

  – If you only need to provide a list of targets, use the following format:

    ```
    targets=database1;database2;database3;
    ```

  – If you only need to provide a list of packs, use the following format:

    ```
    packs=db_diag;db_config;
    ```

  – If you need to provide a list of both targets and packs, use the following format:

    ```
    targets=database1;database2;database3;
    packs=db_diag;db_config;
    ```

- **displayAllMessages**

  Displays all messages. Only error messages are displayed by default. "=value" is not allowed on the command line.

Examples

Example 1 and Example 2 below revoke licenses of specific packs for specific targets. In order to know which target types and pack names you can pass as arguments, you can use the view named mgmt_license_view to see a list of licensable targets, their target types, and the list of packs licensed on them.

To obtain this information, do the following:

1. Access SQL*Plus with your username and password, using sysman or other user that has access to sysman.mgmt_license_view.
2. Select a distinct pack name from sysman.mgmt_license_view, where:

   ```
   target_type=<oracle_database>
   ```

The following example shows pack names for an Oracle database you specify as the target type.

```
PACK_NAME
-------------------
db_config
provisioning
db_sadm
db_tuning
db_diag
```

```
provisioning_db
db_chgmgt

7 rows selected.
```

Based on this information, to revoke a license to the database1 target for the db_chgmgt pack, you would enter the following command:

```
emcli revoke_license_with_validation -type="oracle_database" -targets="database1"
-packs="db_chgmgt"
```

The only limitation of mgmt_license_view is that it only lists the packs for a target type where the pack is granted to at least one target of that type. That is, if the pack is not granted to any target of that type, mgmt_license_view cannot provide any information.

Example 1

The following example revokes the license of the db_diag and db_config packs to database1, database2, and database3 targets (oracle_database target type):

```
emcli revoke_license_with_validation -type="oracle_database"
          -targets="database1;database2;database3;" -packs="db_diag;db_config;"
```

Example 1

The following example revokes the license of the db_diag and db_config packs to database1, database2, and database3 targets (oracle_database target type):

```
emcli revoke_license_with_validation -type="oracle_database"
          -targets="database1;database2;database3;" -packs="db_diag;db_config;"
```

Example 2

The following example revokes the license of the db_diag and db_config packs to all database targets in the setup:

```
emcli revoke_license_with_validation -type="oracle_database"
       -packs="db_diag;db_config;"
```

Example 3

The following example revokes the license of all packs (applicable to database targets) to database1, database2, and database3 targets in the setup:

```
emcli revoke_license_with_validation -type="oracle_database"
         -targets="database1;database2;database3;"
```

Example 4

The following example revokes the license of all packs (applicable to database targets) to all database targets in the setup:

```
emcli revoke_license_with_validation -type="oracle_database"
```

Example 5

The following example uses a text file to pass targets and pack names as the argument. It revokes the license of the db_diag and db_config packs to the database1, database2, and database3 targets (oracle_database target type):

```
emcli revoke_license_with_validation -type="oracle_database"
         -file="/usr/admin1/db_license.txt"
            targets=database1;database2;database3;
```

```
packs=db_diag;db_config;
```

where the content of the "/usr/admin1/license/db_license.txt" file is as follows:

```
targets=database1;database2;database3;
packs=db_diag;db_config;
```

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:

  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 job privileges require specifying jobid:

  VIEW_JOB

  FULL_JOB

Examples

Example 1

For user1, the following example revokes full control of the jobs with ID 923470234ABCDFE23018494753091111, and revokes full control on the target host1.example.com:host:

```
emcli revoke_privs
          -name="user1"
          -privilege="FULL_JOB;923470234ABCDFE23018494753091111"
          -privilege="FULL_TARGET;host1.example.com:host"
```

Example 2

The following example revokes the target privileges from EM Role : Role1:

```
emcli revoke_privs
          -name="Role1"
          -privilege="FULL_TARGET;host1.example.com:host"
```

revoke_roles

Revokes the roles to existing an 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

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


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

run_avail_diag

Runs diagnostics for an availability algorithm for a test-based service. This is mostly useful when the "last calculated" time stamp is running behind the current time and the service status has been unresponsive for some time.

Format

```
emcli run_avail_diag
      -name=<target_name>
      -type=<target_type>
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

Examples

```
emcli run_avail_diag  -name='MyTarget' -type='generic_service'
```

run_promoted_metric_diag

Runs promoted metric diagnostics.

Format

```
emcli run_promoted_metric_diag
        -name=<target_name>
        -type=<target_type>
        -promotedMetricName=<metric_name>
        -promotedColumn=<metric_type>
```

Options

- **name**

  Service target name.

- **type**

  Service target type.

- **promotedMetricName**

  Promoted metric name.

- **promotedColumn**

  Promoted metric type.

Examples

```
emcli run_promoted_metric_diag  -name='MyTarget' -type='generic_service'
-promotedMetricName='metric1' -promotedColumn='Performance'
```

save_masking_script

Saves a masking script already generated to the specified path or file.

Format

```
emcli save_masking_script
     -definition_name=<masking_definition_name>
     [-path=file path]
     [-file=file name]
```

Options

- **definition_name**

  Masking definition name.

- **path**

  Path for the file name to save the masking script. File name is automatically generated. The path and file options are mutually exclusive. Only an absolute path is allowed.

- **file**

  File name to save the masking script. The file name must include the absolute path. Either the path or file option must be specified.

Output

Success or error messages

Examples

Example 1

The following example saves the masking script for the definition named mask_hr_data to the /tmp directory:

```
emcli save_masking_script
       -definition_name=mask_hr_data
       -path=/tmp/
```

Example 2

The following example saves the masking script for the definition named mask_hr_data to /tmp/abc.sql :

```
emcli save_masking_script
       -definition_name=mask_hr_data
       -file=/tmp/abc.sql
```

secure_agent

Re-secures up to 50 Grid Control Management Agents after you create a new Certificate Authority for the Grid Control Management Servers.

Agents can be secured by providing a list of Agent names, a group name, or with an input file. If you provide a group name, Enterprise Manager resolves that to a list of Agents that monitor targets in the group. You can also provide an Agent list with an input file to this EMCLI command. For all of these options, you must either provide a username/password, or the user must have been configured with preferred credentials on Agent targets.

The verb submits a job with the list of Agents and the credntials provided as input, and ouputs the Job Name and Job ID, which can be used track the status of the job. This verb also calculates the list of Agents to resecure by filtering out invalid Agents, insecure Agents, Agents that are down, and Agents that already have an active job execution. This verb also filters out Agents that are already secured by the correct Certifcate Authority, but you can disable this filter by using the `-disable_ca_check` option.

Note the following additional points about this verb:

- Does not secure Agents that are not already secured. If you want to secure Agents not currently secured, you need to execute the command "`emctl secure agent`" from the Grid Control Management Agent in `$ORACLE_HOME/bin`.
- Removes down Agents from the list of Agents to be re-secured. emcli requires the Agent to be up and running.
- Removes down Agents from the list of Agents to be re-secured. emcli requires the Agent to be up and running.
- Removes invalid Agents from the list of Agents to be re-secured.
- Removes duplicate Aagents from the list of Agents to be re-secured if an Agent has been provided several times in the list.
- Re-secures up to 50 Agents at one time. If you specify a list of more than 50 Agents to be re-secured, emcli just re-secures the first 50 Agents from the list and ignores the Agents with a rank greater than 50 in the list.
- Filters out an Agent that is already busy running a job execution.
- Filters out an Agent to be re-secured if the Agent is already secured with the latest Certificate. However, you can remove this restriction with the `-disable_ca_check` flag.
- Needs to connect to the Grid Control Management Agent with either the preferred credentials set at the Agent level in the Grid Control repository, or with the user name and password provided.
- If you execute this command with the option `-use_pref_creds`, emcli filters out from the list of Agents to be re-secured all of the Agents with preferred credentials not set or incorrectly set.
- If you execute this command with the `-username` and `-password` options, emcli filters out from the list of Agents to be re-secured all of the Agents that cannot be contacted with the user name and password provided.

Format

```
emcli secure_agent
```

```
[-agt_names="agt1;agt2;..."] [-agt_names_file="<file>"]
[-group_name="group_name"]
[-use_pref_creds]
[-username="username"]
[-password="password"]
[-disable_ca_check]
```

Options

- **agt_names**

  Semicolon-separated list of Agent names.

- **agt_names_file**

  Absolute path of the file containing the list of Agent names, each on a new line.

- **group_name**

  Identifies the list of Agents to secure. Enterprise Manager resolves the list of Agents that monitor (not just members of the group) the list of targets in the group.

- **use_pref_creds**

  Use preferred credentials configured for the Agent to execute the secureAgent job.

- **username**

  User name to execute the secureAgent job at the Agent.

- **password**

  Password to execute the secureAgent job at the Agent.

- **disable_ca_check**

  Flag to disable the check to verify if the Agents are secured with the latest Certificate Authority.

Examples

The following example assumes that the Grid Control Management Agents are installed with the user `oracle` on all hosts with the same password.

```
emcli secure_agent -agt_names="ushost1.mycompany.com:3872;
ushost2.mycompany.com:3872;ushost1.mycompany.com:3872" -username="oracle"
-password="mypwd"
Summary :
Number of valid agents provided for secure : 2, Filtered : 0, Selected : 2
Details :
Number of agents provided for secure (input) : 3
After merging and removing invalid/duplicate agents : 2

Job "SECUREAGENTS JOB 2010-Dec-15 16:08:45" submitted for securing 2 agents
Job ID: DE404669C52F43AC805151306F4B138D
Execution ID: 47225EF8C9A8455FBDFB817BF8722FFF
```

The following example re-secures the Agents, even if they are already registered to the latest certificate.

```
emcli secure_agent
-agt_names="ushost1.mycompany.com:3872;ushost2.mycompany.com:3872;
ushost3.mycompany.com:3872" -username="oracle" -disable_ca_check
```

set_agent_property

Modifies a specific Agent property. You can use this command if you have operator privilege for the Agent.

Format

```
emcli set_agent_property
      -agent_name="<agent_target_name>"
      -name="<agent_property_name>"
      -value="<agent_property_value>"
```

Options

- **agent_name**

  Name of the Agent target.

- name

  Name of the Agent property you want to modify.

- value

  New value for the Agent property.

Examples

The following example sets the value of the UploadInterval property to 15.

```
emcli get_agent_property -agent_name="agent.example.com:11850"
         -name=UploadInterval
         -value=15
```

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

Example 1

The following example sets the availability of the service MyTarget to be based on all key tests:

```
emcli set_availability -name='MyTarget' type='generic_service'
                       -availType='test' -availOp='and'
```

Example 2

The following example sets the availability of the service MyTarget to be based on any key test:

```
emcli set_availability -name='MyTarget' type='generic_service'
                       -availType='test' -availOp='or'
```

set_credential

Sets preferred credentials for given users.

Format

```
emcli set_credential
      -target_type="ttype"
      [-target_name="tname"]
      -credential_set="cred_set"
      [-user="user"]
      -columns="col1:newval1;col2:newval2;PDP:SUDO/POWERBROKER;RUNAS:oracle;
         PROFILE:user1..."
      [-input_file="tag1:file_path1;tag2:file_path2;..."]
      [-oracle_homes="home1;home2"]
      [-monitoring]

[ ] denotes that the parameter is optional
```

Options

- **target_type**

  Type of target. The must be "host" if 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 if 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 the 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` parameter. You can specify this option 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.

- **monitoring**

  Flag indicating that credentials affected are monitoring credentials. If omitted, the credentials affected are preferred credentials. Monitoring credentials require specifying the target_name option.

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.example.com
      -credential_set=OHCreds
      -user=admin1
      -column="OHUsername:joe;OHPassword:newPass"
      -oracle_homes="database1;mydb"
```

set_instance_jobgrants

Defines key beacons and tests of the service.

Format

```
emcli set_instance_jobgrants
      -instance_guid=<instance guid>
      -grants=<user:privilege>
```

Options

- **instance_guid**

  GUID of the instance.

- **grants**

  String of user:privilege pairs each separated by a semi-colon (;) , where:

  ```
  user = em user name
  privilege  = VIEW_JOB or FULL_JOB
  ```

Examples

```
emcli set_instance_jobgrants -instance_guid=16B15CB29C3F9E6CE040578C96093F61
-grants="user1:VIEW_JOB;user2:FULL_JOB"
```

set_key_beacons_tests

Defines key beacons and tests of the service.

Format

```
emcli 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

Example 1

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
```

Example 2

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

```
emcli set_metric_promotion
      -name=Service target name
      -type=Service target type
      [-category = Usage/Performance/Business]
      -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, performance, or business 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 to 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

Example 1

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
```

Example 2

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
```

Example 3

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

```
emcli 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 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

Example 1

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'
```

Example 2

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'
```

set_standby_agent

Permits targets to relocate from one agent to another. This verb always populates a table that determines which targets from the source agent to the destination agent are permitted to relocate for the Enterprise Manager target.

Format

```
emcli set_standby_agent
       -src_agent=<source agent>
       -dest_agent=<destination agent>
       -target_name=<target name>
       -target_type=<target type>

[ ] denotes that the parameter is optional
```

Options

- **src_agent**

  Agent currently monitoring the targets. If srcAgent is not known, enter currentOwner as argument

- **dest_agent**

  Agent you want to monitor the targets.

- **target_name**

  Name of the to be moved.

- **target_type**

  Type of target to be moved.

Output

Output message of the command execution.

set_target_property_value

Sets the value of a target property for a specified target. Any prior values of the target property are overwritten. When assigning values to the Oracle-provided target properties, use the English names of these target properties:

Comment, Deployment Type, Line of Business, Location, Contact

Format

```
emcli set_target_property_value
      -property_records="target_name:target_type:property_name:property_value"
      [-separator=property_records="sep_string"]
      [-subseparator=property_records="subsep_string"]
      [-input_file="parameter_tag:file_path"]

[ ] denotes that the parameter is optional
```

Options

- **property_records**

  List of property records. The following parts comprise each property record:

  <target_name>:<target_type>:<property_name>:property_value>

  - target_name — Target name of the target for which you want to update the property.
  - target_type — Target type of the target.
  - property_name — Name of the property whose value you want to update. Property names are case sensitive.
  - property_value — Value to be assigned/updated for the property.

- **separator**

  When specifying multiple property records, use the separator string delimiter as a delimiter between property records. The default separator delimiter is ";".

- **subseperator**

  String delimiter to be used between parts of a property record. The default subseparator delimiter is ":".

- **input_file**

  Used in conjunction with the -property_records option, this option enables you to provide the property records in a file. This option specifies a mapping between a tag and a local file path. The tag is specified in lieu of property records. The tag cannot contain colons (:) or semi-colons (;) .

Examples

Example 1

The following example sets the 'Owner Name' property to Jane Smith for the database test_database.

```
emcli set_target_property_value
      -property_records="test_database:oracle_database:Owner Name:Jane Smith"
```

Example 2

The following example sets the Owner property to Jane Smith for the database test_db and also sets the Asset Number property to 100 for the database test_db1.

```
emcli set_target_property_value
          -property_records="test_db:oracle_database:Owner:Jane Smith;
           test_db1:oracle_database:Asset Number:100"
```

Example 3

The following example takes the input of the property records from the specified file /temp/rec_file.

```
emcli set_target_property_value
          -property_records="REC_FILE" -input_file="REC_FILE:/temp/rec_file"
```

The file /temp/rec_file would contain entries such as:

```
test_db:oracle_database:Owner:Jane Smith;test_db1:oracle_database:Asset Number:100
```

Example 4

The following example sets the Owner property to Jane Smith for the test_db database, and sets the Asset Number property to 100 for the test_db1 database. The separator used within the records is "," and the subseparator is "@" .

```
emcli set_target_property_value
           -property_records="test_db@oracle_database@Owner@
            Jane Smith,test_db1@oracle_database@AssetNumber@100"
```

set_test_threshold

Sets a test threshold.

Format

```
emcli set_test_threshold
      -name=<target_name>
      -type=<target_type>
      -testname=<test_name>
      -testtype=<test_type>
      -metricName=<metric_name>
      -metricColumn=<metric_column>
      -occurrences=<occurrences>
      [-warningThres=<warning_threshold>]
      [-criticalThres=<critical_threshold>]
      [-operator=<operator>]
      [-beaconName=<beacon_name>]
      [-stepName=<step_name>]
      [-stepGroupName=<stepgroup_name>]

[ ] denotes that the parameter is optional
```

Examples

```
emcli set_test_threshold -name="Service Name"
              -type="generic_service"
              -testname="Test Name"
              -testtype="HTTP"
              -metricName="http_response"
              -metricColumn="timing"
              -occurrences=1
              -warningThres=100000
```

setup

Configures EM CLI to work with a specific management server.

The default mode stores the credentials, which is inherently insecure because of backward compatibility reasons. For a secure setup, you need to specify the noautologin option. See noautologin in the Options section for more information.

Format

```
emcli setup
      -url="http[s]://host:port/em/"
      -username=<EM Console Username>
      [-password=<EM Console Password>]
      [-ssousername=<EM SSO Username>]
      [-ssopassword=<EM SSO Password>]
      [-licans=yes|no]
      [-dir=<local emcli configuration directory>]
      [-trustall]
      [-novalidate]
      [-noautologin]
      [-custom_attrib_file=<Custom attribute file path>]
      [-nocertvalidate]

[ ] 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. (https is recommended for security reasons).

- **username**

  Enterprise Manager user name to be used by all subsequent EM CLI commands when contacting the OMS.

  If the SSO user is also an Enterprise Manager user (that is, authenticated in LDAP/OID), you can only register EM CLI with the ssousername option. After you enable SSO for the OMS, you cannot subsequently register EM CLI with only username.

- **password**

  Enterprise Manager user password. If you do not specify this option, you are prompted for the password interactively.

  Note: Providing a password on the command line is insecure and should be avoided.

- **ssousername**

  SSO user name to be used by all subsequent EM CLI commands when contacting the OMS. Its value must be provided only if Enterprise Manager is configured to use SSO.

  If the SSO user is also an Enterprise Manager user (that is, authenticated in LDAP/OID), you can only register EM CLI with this option. After you enable SSO

for the OMS, you cannot subsequently register EM CLI with only the username option.

You need to specify username and ssousername together if the OMS is protected by SSO, and ssousername is not an Enterprise Manager user, but instead a standard LDAP user, such as orcladmin. For example:

```
emcli setup -username=myusername -ssousername=myssouser
-url=https://abc.example.com:4566/em/
```

- **ssopassword**

  SSO user password. If you do not specify this option, you are prompted for the password interactively.

  Note: Providing a password on the command line is insecure and should be avoided.

- **licans**

  Indicates whether the license is accepted or not accepted by the user. Specify yes to accept the license, or specify no to not accept the license.

- **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 accepts any server certificate from the OMS, which results in lower security.

- **novalidate**

  Does not authenticate the Enterprise Manager user name or SSO user name against the OMS. Assume the given user name is valid.

- **noautologin**

  Set up the emcli client in this mode. If used, secure mode is implemented, which does not store any Enterprise Manager or SSO password on the client disk. The credentials are provided once at the time of setup after which a session is established between the client and OMS. All the subsequent verbs will use this session. Inactivity or an explicit logout (using the Logout verb) terminates this session, and a re-setup or explicit login (using the Login verb) is required before invoking any new verb.

  In the default case, you do not need to explicitly log in when the session expires. Enterprise Manager user or SSO credentials are stored on the client system, and the client implicitly logs in again using these credentials when the session expires. No user intervention is required in case of session expiration.

- **custom_attrib_file**

  Path name of a file containing Audit Custom Attribute values. This option is required when the OMS is configured for Audit Custom Attributes. If you do not

provide custom_attrib_file, you are prompted to enter the values of the custom attributes.

The file can contain up to three lines, each containing the description of one custom attribute. Each line should be of the form:

```
<attr-name>#<attr-displayname>#<isMandatory>#<attr-value>
```

- **#** — Field separator.
- **attr-name** — Name of the attribute.
- **attr-displayname** — Display name of the attribute.
- **isMandatory** — 1 if the attribute is mandatory, otherwise 0.
- **attr-value** — Value of the custom attribute.

- **nocertvalidate**

  Does not validate the host name in the SSL Certificate that the OMS provides.

Examples

```
emcli setup -url=http://myworkstation.example.com:7770/em -username=sysman
```

To configure the EM CLI Client to function with multiple OMSs by implementing multiple setups, do the following:

1. Set up the EM CLI client for OMS1 at location dir1:

   ```
   emcli setup -dir=<dir1> -url=<Url of OMS1> -user=<EM Username for OMS1>
   ```

2. Set up the EM CLI client for OMS2 at location dir2:

   ```
   emcli setup -dir=<dir2> -url=<Url of OMS1> -user=<EM Username for OMS2>
   ```

3. Set the environment variable EMCLI_STATE_DIR to point to the setup directory for OMS1:

   ```
   setenv EMCLI_STATE_DIR <dir1>
   ```

   This sets the EM CLI Client to function with OMS1.

4. Set the environment variable EMCLI_STATE_DIR to point to the setup directory for OMS2:

   ```
   setenv EMCLI_STATE_DIR <dir2>
   ```

   This sets the EM CLI Client to function with OMS2.

show_audit_settings

Shows the following details of the current audit settings:

— Audit Switch
— Externalization Switch
— Directory
— File Prefix
— File Size
— Data Retention Period

Format

```
emcli show_audit_settings
```

show_credential_set_info

Displays the parameters of credential sets defined with target types.

Format

```
emcli show_credential_set_info
       [-target_type="<target_type>"]
       [-set_name="<credential_set_name>"]

[ ] denotes that the parameter is optional
```

Options

- **target_type**

  Type of target. The default is to display the credential set defined for all target types.

- **set_name**

  Name of the credential set. The default is to display all credential sets defined for a target type.

Examples

Example 1

The following example displays the details of all credential sets defined with all target types:

```
emcli show_credential_set_info
```

Example 2

The following example displays all credential sets defined with the oracle_database target type:

```
emcli show_credential_set_info -target_type=oracle_database
```

Example 3

The following example displays the details of the HostUDMCreds credential set defined for the host target type.

```
emcli show_credential_set_info -target_type=host
      -set_name=HostUDMCreds
```

show_credential_type_info

Displays the parameters of credential types defined for target types.

Format

```
emcli show_credential_type_info
       [-target_type="<target_type>"]
       [-type_name="<credential_type_name>"]

[ ] denotes that the parameter is optional
```

Options

- **target_type**

  Type of target. The default is to display the credential set defined for all target types.

- **type_name**

  Name of the credential type. The default is to display all credential types defined for a target type.

Examples

Example 1

The following example displays the details of all credential types defined with all target types:

```
emcli show_credential_type_info
```

Example 2

The following example displays all credential types defined with the oracle_database target type:

```
emcli show_credential_type_info -target_type=oracle_database
```

Example 3

The following example displays the details of the HostUDMCreds credential type defined for the oracle_database target type.

```
emcli show_credential_type_info -target_type=oracle_database
      -type_name=HostUDMCreds
```

show_operations_list

Shows the list of all auditable Enterprise Manager operations names.

Format

```
emcli show_operations_list
```

Ouput

Output appears as shown in the following example:

```
ADD_AGENT_REGISTRATION_PASSWORD
AGENT_REGISTRATION_PASSWORD_USAGE
AGENT_RESYNC
APPLY_TEMPLATE
AUDIT_EXPORT_SETTINGS
AUDIT_SETTINGS
CHANGE_PASSWORD
CHANGE_PREFERRED_CREDENTIAL
CREATE_PG_SCHED
CREATE_ROLE
CREATE_TEMPLATE
CREATE_UDP
CREATE_UDPG
CREATE_USER
DELETE_AGENT_REGISTRATION_PASSWORD
DELETE_JOB
DELETE_PG_EVAL
DELETE_PG_SCHED
DELETE_ROLE
DELETE_TEMPLATE
DELETE_UDP
DELETE_UDPG
DELETE_USER
EDIT_AGENT_REGISTRATION_PASSWORD
EDIT_JOB
EDIT_PG_SCHED
EDIT_TEMPLATE
EDIT_UDP
EDIT_UDPG
EVALUATE_UDP
FILE_TRANSFER
GET_FILE
GRANT_JOB_PRIVILEGE
GRANT_ROLE
GRANT_SYSTEM_PRIVILEGE
GRANT_TARGET_PRIVILEGE
IMPORT_UDP
JOB_OUTPUT
LOGIN
LOGOUT
MODIFY_METRIC_SETTINGS
MODIFY_POLICY_SETTINGS
MODIFY_ROLE
MODIFY_USER
PUT_FILE
REMOTE_OPERATION_JOB
REMOVE_PRIVILEGE_DELEGATION_SETTING
REPOSITORY_RESYNC
```

```
REVOKE_JOB_PRIVILEGE
REVOKE_ROLE
REVOKE_SYSTEM_PRIVILEGE
REVOKE_TARGET_PRIVILEGE
SAVE_MONITORING_SETTINGS
SET_PRIVILEGE_DELEGATION_SETTING
SUBMIT_JOB
SUSPEND_JOB
```

start_guest_vm

Starts a guest virtual machine. To start the guest virtual machine, it should be in the Halted state.

Format

```
emcli start_guest_vm
      -guest_vm_name=<Virtual Machine Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- server_pool_name

  Name of the server pool.

Examples

The following example starts the dom15 guest Virtual Machine.

```
emcli start_guest_vm -guest_vm_name="dom15" -server_pool_name="Oracle Server Pool"
```

start_paf_daemon

Starts the Deployment Procedure Manager Daemon.

Format

```
emcli start_paf_daemon -interval=<number in minutes>
```

Options

- **interval**

  Number in minutes that the Deployment Procedure Manager Daemon should wait between each run.

start_vt_daemon

Starts the virtualization daemon.

Tip: See also stop_vt_daemon on page 2-266.

Format

```
emcli start_vt_daemon
```

Options

None.

status_paf_daemon

Gets the Deployment Procedure Manager Daemon status.

Format

```
emcli status_paf_daemon
```

Options

None.

status_vt_daemon

Gets the status of the virtualization daemon.

Format

```
emcli status_vt_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

```
emcli stop_blackout
      -name="name"
      [-createdby="blackout_creator"]

[ ] denotes that the parameter is optional
```

Options

- **name**

  Name of the blackout to stop.

- **createdby**

  Enterprise Manager user who created the blackout. The default is the current user. The SUPER_USER privilege is required to stop a blackout created by another user.

Examples

Example 1

The following example stops blackout backup_db3 created by the current user.

```
emcli stop_blackout -name=backup_db3
```

Example 2

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_guest_vm

Stops a guest Virtual Machine. To stop the guest Virtual Machine, it should be in the Running state.

Format

```
emcli stop_guest_vm
      -guest_vm_name=<Virtual Machine Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- **server_pool_name**

  Name of the server pool.

Examples

The following example stops the dom15 guest Virtual Machine.

```
emcli stop_guest_vm -guest_vm_name="dom15" -server_pool_name="Oracle Server Pool"
```

stop_instance

Stops a scheduled, failed, or running deployment instance.

Format

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

Options

- **instance**

  GUID of the instance.

Examples

```
emcli stop_instance -instance=16B15CB29C3F9E6CE040578C96093F61
```

stop_job

Stops a specified job. You can use the `get_jobs` verb to obtain a list of job IDs and names.

Format

```
emcli 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

Example 1

The following example stops a job with the specified ID.

```
emcli stop_job -job_id=12345678901234567890123456789012
```

Example 2

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.

stop_virtual_server

Stops a virtual server. To stop the virtual server, it should be in the Up state.

Format

```
emcli stop_virtual_server
      -server_name=Server name
```

Options

- **server_name**

  Name of the virtual server.

Examples

The following example stops the st-vs1.example.com virtual server.

```
emcli stop_virtual_server -server_name="st-vs1.example.com"
```

stop_vt_daemon

Stops the virtualization daemon.

Tip: See also start_vt_daemon on page 2-257.

Format

```
emcli stop_vt_daemon
```

Options

None.

submit_agent_patch

Patches the Agent. All of the inputs should be present in the targets_file.xml file.

Format

```
emcli submit_agent_patch
-input_file="data:targets_file.xml"
-schedule="start_time:<value>;tz:<value>;grace_period:<value>"
```

Options

- **input_file**

  XML file name containing the necessary input.

- **schedule**

  Specify the starting time, time zone, and grace period.

Examples

```
emcli submit_agent_patch -input_file="data:targets_file.xml"
-schedule="start_time:2006/6/21 21:23;tz:America/New_York;grace_period:15"
```

submit_job

Creates and submits a job.

Format

```
emcli submit_job
      -job="name:type"
      -targets="name1:type1;name2:type2;..."
      -parameters="name1:value1;name2:value2;PDP:SUDO/POWERBROKER;RUNAS:oracle;
         PROFILE:user1..."
      [-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]]
            [tzregion:<...>]
            [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 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 considered. 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=tzoffset**

  Value of the timezone. When you do not specify the `tzinfo` argument or it is `repository`, the `timezone` value is the repository time zone. In this case, you cannot 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 time zone. 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 time zone is applied to the start time and the end time of the job schedule. The time zones associated with each target are not considered when scheduling the job (except that when you set `tzinfo` to `target`, the specified target's timezone is used for the job schedule).

- **schedule=tzregion**

  Time zone region to use. When you specify the tzinfo option, this argument determines which time zone to use for the job schedule. Otherwise, it is ignored. The default is GMT.

- **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.

- **noheader**

  Displays 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.

  – format=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

  – row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Output Columns

Job ID, Execution ID

Examples

Example 1

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.example.com:host"
```

Example 2

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"
```

Example 3

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_masking_job

Submits a masking job and returns the display job ID and execution ID.

Format

```
emcli submit_masking_job
  -definition_name=masking_defn_name
  [-seed=seed_string]
  [-credential_set_name=credential_set_name]
  [-parameters=name1:value1;name2:value2;...]
  [-script_file_location=script_file_location]
  [-script_file_name=script_file_name]
  [-input_file=PWD_FILE_TAG:credentials file name]
  [-script | -format=[name:<pretty|script|csv>];
                     [column_separator:"column_sep_string"];
                     [row_separator:"row_sep_string"];
  ]

[ ] denotes that the parameter is optional
```

Options

- **definition_name**

  Masking definition name.

- seed

  Seed string must be specified if the masking definition involves usage of substitute format.

- **credential_set_name**

  Set name of the preferred credentials stored in the Enterprise Manager repository. You are only required to specify the DBCreds set. The host credentials set is taken as DBHostCreds. The valid values for this parameter are:

  – **DBHostCreds** —

    Host credential set for an oracle_database target. This is the default host credential set and need not be specified.

  – **DBCredsNormal** —

    Default normal credential set for an oracle_database target. If no value is specified, DBCredsNormal is used.

  – **DBCredsSYSDBA** —

    SYSDBA credential set for an oracle_database target.

  You can only specify this parameter when the override credential parameters such as "[db | host]_username" and "[db | host]_password" are not present.

- **parameters**

  List of name-value pairs that represent the credentials required for connecting to the database instance. The supported parameters are db_username, db_password, db_role, host_username, and host_password.

- script_file_location

Location where the SQL script is to be copied and executed. Default values of $ORACLE_HOME/dbs are used if a value is not specified.

- script_file_location

  Location where the SQL script is to be copied and executed. Default values of $ORACLE_HOME/dbs are used if a value is not specified.

- **script_file_name**

  Name of the script file to store the masking SQL script. If you do not specify a name, a system generated file name is used.

- **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.

  – `format`=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

  – row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Examples

Example 1

The following example submits a masking job for the definition name email1 and returns the job ID and execution ID:

```
emcli submit_masking_job -definition_name=email1 -parameters="db_username:sys;
    db_password:password;db_role:SYSDBA;host_username:test;host_password:password"
```

Example 2

The following example assumes the default credential set as DBCredsNormal and returns job ID and execution ID.

```
emcli submit_masking_job -definition_name=email1
```

Example 3

The following example picks up credentials from the files host_creds.txt and db_creds.txt.

```
emcli submit_masking_job -definition_name=email1 -parameters="HOST_CREDS;DB_CREDS"
-input_file=HOST_CREDS:host_creds.txt -input_file=DB_CREDS:db_creds.txt
```

It is also possible to specify both of the credentials in one file and use only one -input_file tag. If PDP must be used, you need to provided values in the parameters/input_file as follows:

- SUDO:

  ```
  db_username:sys;db_password:password;db_role:SYSDBA;host_username:user2;host_
  password:password;PDP:SUDO;RUNAS:user1
  ```

  POWERBROKER:

  ```
  db_username:sys;db_password:password;db_role:SYSDBA;host_username:user2;host_
  password:password;PDP:POWERBROKER;RUNAS:user1;PROFILE:profile
  ```

Example 4

The following example uses the database credential set name as DBCredsSYSDBA, and assumes the database host credential set name as DBHostCreds. If the masking definition involves usage of a substitute format, it uses the seed string of 'abcd'. The example also overrides the default script file name and location by the values specified. The example aslo submits a masking job for the given definition name and returns job ID and execution ID.

```
emcli submit_masking_job -definition_name=email2
-credential_set_name=DBCredsSYSDBA -seed=abcd -script_file_location=/tmp
-script_file_name=email1.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**

  Input data for the Deployment Procedure. The `file_path` should point to a file containing the data XML file.

- **instance_name**

  Name of the procedure instance.

- **schedule**

  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

```
emcli 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

Example 1

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
```

Example 2

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 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"
```

suspend_guest_vm

Suspends a guest Virtual Machine. To suspend the guest Virtual Machine, it should be in the Running state.

Tip: See also resume_guest_vm on page 2-214.

Format

```
emcli suspend_guest_vm
      -guest_vm_name=<Virtual Machine Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- **server_pool_name**

  Name of the server pool.

Examples

The following example suspends the dom15 guest Virtual Machine.

```
emcli suspend_guest_vm -guest_vm_name="dom15" -server_pool_name="Oracle Server
Pool"
```

suspend_instance

Suspends a running deployment instance.

Format

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

Options

- **instance**

  GUID of the instance.

Examples

```
emcli suspend_instance -instance=16B15CB29C3F9E6CE040578C96093F61
```

sync

Synchronizes the EM CLI client with an OMS. After synchronization, all verbs and associated command line help available to this OMS become available at the EM CLI client.

Synchronization occurs automatically during a call to setup.

Format

```
emcli sync
```

Options

None.

Examples

```
emcli sync
```

sync_beacon

Synchronizes a beacon that is monitoring the target (reloads all collections to beacon).

Format

```
emcli 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 the `MyTarget` target of type `generic_service`.

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

unpause_guest_vm

Unpauses a paused guest virtual machine. To unpause the guest virtual machine, it should be in the Paused state.

Tip: See also pause_guest_vm on page 2-200.

Format

```
emcli unpause_guest_vm
      -guest_vm_name=<Virtual Machine Name>
      -server_pool_name=<Server Pool Name>
```

Options

- **guest_vm_name**

  Name of the guest Virtual Machine.

- **server_pool**

  Name of the guest server pool.

Examples

The following example unpauses the paused dom15 guest Virtual Machine.

```
emcli pause_guest_vm -guest_vm_name="dom15" -server_pool_name="Oracle Server Pool"
```

update_audit_settings

Updates the current audit settings in the repository and restarts the OMS.

Format

```
emcli update_audit_settings
     [-externalization_switch="ENABLE/DISABLE"]
     -directory_name="<database_directory_name>"
     -file_prefix="<file_prefix>"
     -file_size="<file_size (Bytes)>"
     -data_retention_period="<data_retention_period (Days)>"

[ ] denotes that the parameter is optional
```

Options

- **externalization_switch**

  Enable the audit data export service. The default value is DISABLE.

- **directory_name**

  Database directory that should be configured with an OS directory where the export service archives the audit data files.

- **file_prefix**

  File prefix to be used by export service to create the file name where audit data is to be written. The default value is em_audit.

- **file_size**

  Maximum value of each file size. The default value of this option is 5000000 bytes.

- **data_retention_period**

  Maximum period the Enterprise Manager repository stores audit data. The default value is 365 days.

Examples

Example 1

The following example enables all operations except LOGIN and LOGOUT:

```
emcli update_audit_settings
          -audit_switch="ENABLE"
          -operations_to_enable="ALL"
          -operations_to_disable="LOGIN;LOGOUT"
```

Example 2

```
emcli update_audit_settings
    -externalization_switch="ENABLE"
    -directory="EM_DIR"
    -file_prefix="my_audit"
    -file_size="10000"
    -data_retention_period="60"
```

update_and_retry_step

Updates arguments of the failed step and retries it.

Format

```
emcli update_and_retry_step
      -instance=instance_guid
      -stateguid=state_guid
      [-args="command1:value1;command2:value2;..."]

[ ] denotes that the parameter is optional
```

Options

- **instance**

  GUID of the instance.

- **stateguid**

  State GUID.

- **args**

  Arguments of the step to be updated during retry. For the full list of arguments that can be updated, see the get_retry_arguments verb.

Examples

```
emcli update_and_retry_step -instance=16B15CB29C3F9E6CE040578C96093F61
-stateguid=51F762417C4943DEE040578C4E087168 -args="command:ls"
```

update_db_password

Updates the target database password change in the Enterprise Manager Credential sub-system and can change the password on the target database as well. This verb also propagates the collection or monitoring credentials to Enterprise Manager Agents.

Format

```
emcli update_db_password
       -target_name="tname"
       -user_name="user_name"
       [-target_type="ttype"]
       [-change_all_references="yes/no"]
       [-change_at_target="yes/no"]
       [-input_file="tag1:file_path1;tag2:file_path2;..."]

[ ] denotes that the parameter is optional
```

Options

- **target_name**

  Name of the target.

- **user_name**

  Name of the database user.

- **target_type**

  Type of target. The possible values for target type in this verb are -oracle_database and - rac_database. The default value for this parameter is oracle_database.

- **change_all_references**

  Specify if the password must be changed for all references in Enterprise Manager. Possible values are:

  – **yes** — Update all password references in Enterprise Manager for a DBSNMP user who has an old password that matches the new password.

  – **no** — Update the password for the currently logged in user.

  The default value of this option is Yes.

- **change_at_target**

  Specify whether the password must also be changed on the target. This option is not supported for a SYS user.

  – **yes** — Change the password on the target database.

  – **no** — Update the password only on Enterprise Manager.

  The default value of this option is No.

- **input_file**

  Path of the file that has old and new passwords. Use this option to hide passwords displayed on the command line. You must accompany each path with a tag referenced in the password options.

  When you execute this verb with the input_file option, you are prompted to enter the following values in non-echo mode:

-old_password
-new_password
-retype_new_password

Examples

```
emcli update_db_password
      -target_name=myDB
      -user_name=Admin1

emcli update_db_password
      -target_name=myDB
      -user_name=Admin1
      -change_at_target=yes
```

update_host_password

Updates the changed host password in the credential sub-system. For collection or monitoring credentials, the password change is also propagated to the Enterprise Manager Agent.

Format

```
emcli update_host_password
      -target_name="tname"
      -user_name="user_name"
      [-change_all_references="yes/no"]
      [-input_file="tag1:file_path1;tag2:file_path2;..."]

[ ] denotes that the parameter is optional
```

Note: When you execute this verb, you are prompted to enter the following values in non-echo mode:

-old_password
-new_password
-retype_new_password

Options

- **target_name**

  Name of the target.

- **user_name**

  Name of the database user.

- **change_all_references**

  Specifies if the password must be changed for all references in Enterprise Manager for the given user.

  Possible values are:

  – Yes — Updates all references in Enterprise Manager for this password.

  – No — Updates the password for the current logged-in user. This is the default.

- **input_file**

  File path that has old and new passwords. This option hides passwords. You must accompany each path with a tag referenced in the password options.

Examples

Example 1

The following example asks the user to enter the values of the old and new passwords, then retype the new password to update the new password in Enterprise Manager for this target reference.

```
emcli update_host_password
      -target_name=myHost
      -user_name=Admin1
```

Example 2

The following example asks the user to enter the values of the old and new passwords, then retype the new password to update the new password in Enterprise Manager for all users' credentials referenced with the myHost target name and Admin1 user name.

```
emcli update_host_password
      -target_name=myHost
      -user_name=Admin1
      -change_all_references=yes
```

update_password

Updates passwords (or other credentials) for a given target.

Format

```
emcli 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 available for hiding 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>
```

Example 1

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 user name. 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
```

Example 2

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
```

update_target_password

Updates the changed target password in the Enterprise Manager credential sub-system. For collection or monitoring credentials, the password change is also propagated to Enterprise Manager agents.

Format

```
emcli update_target_password
        -target_type="ttype"
        -target_name="tname"
        -key_column="column_name:column_value"
        [-change_all_references="yes/no"]
        [-input_file="tag1:file_path1;tag2:file_path2;..."]

[ ] denotes that the parameter is optional
```

Note: When you execute this verb, you are prompted to enter the following values in non-echo mode:

-old_password
-new_password
-retype_new_password

Options

- **target_type**

  Type of target.

- **target_name**

  Name of the target.

- **key_column**

  Name and value of the key column for the credential type. The key column usually represents the user name.

  To obtain the key column for a target type, enter the following command:

  ```
  emcli get_credential_type_info -target_type=<target_type>"
  ```

  To obtain the key column for all target types, enter the following command:

  ```
  emcli get_credential_type_info
  ```

- **change_all_references**

  Specifies if the password must be changed for all references in Enterprise Manager for the given user.

  Possible values are:

  – Yes — Updates all references in Enterprise Manager for this password.

  – No — Updates the password for the current logged-in user. This is the default.

- **input_file**

  File path that has old and new passwords. This option hides passwords. You must accompany each path with a tag referenced in the password options. You can specify this option more than once.

Examples

Example 1

The following example asks the user to enter the values of the old and new passwords, then retype the new password to update the new password in Enterprise Manager for this target reference.

```
emcli update_target_password
      -target_type=host
      -target_name=myHost
      -key_column=HostUserName:Admin1
```

Example 2

The following example asks the user to enter the values of the old and new passwords, then retype the new password to update the new password in Enterprise Manager for all users' credentials referenced with the mydb target name and Admin1 user name.

```
emcli update_target_password
      -target_type=oracle_database
      -target_name=mydb
      -key_column=DBUserName:Admin1
      -change_all_references=yes
```

version

Lists EM CLI verb versions or the EM CLI client version.

Format

```
emcli version
    [-verb_name=<verb_name_filter>]
    [-exact_match]
    [-noheader]
    [-script | -format=
               [name:"pretty|script|csv"];
               [column_separator:"column_sep_string"];
               [row_separator:"row_sep_string"];
    ]

[ ] denotes that the parameter is optional
```

Options

- **verb_name**

  Verb name filter. Selects matching EM CLI verb names. When you specify this option, an output table shows the version for each verb whose name matches <verb_name_filter>. The EM CLI client version is displayed when you do not specify this option.

  Verb filters use regular expression pattern matching unless you specify -exact_match. A zero length filter matches everything.

  Note: For Unix csh, use single quotes around a filter value containing '$' .

- exact_match

  Uses exact matching for filters.

- noheader

  Displays 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.

  – `format`=column_separator:"column_sep_string" column-separates the Verb output by <column_sep_string>. Rows are separated by the newline character.

– row_separator:"row_sep_string" row-separates the Verb output by <row_sep_string>. Rows are separated by the tab character.

Output Columns

Verb, Version (when -verb_name is specified)

Examples

Example 1

The following example shows the version for all verbs:

```
emcli version -verb_name=
```

Example 2

The following example shows the version for all verbs with names that exactly match the string "sync" :

```
emcli version -verb_name=sync -exact_match
```

Example 3

The following example shows the version for all verbs with names starting with "log:"

```
emcli version -verb_name="^log"
```

Example 4

The following example shows the version for all verbs with names that end with "in:"

```
emcli version -verb_name="in$"
```

Example 5

The following example shows the version for all verbs with names that contain a substring matching "elp" or with names that begin with "ver" or "lo", contains "i", and ends with "n:"

```
emcli version -verb_name="elp|^(ver|lo).*i.*n$"
```

Example 6

The following example shows the version for all verbs with names that exactly match the string "setup." Alternatively, you could use the -exact_match option.

```
emcli version -verb_name="^setup$"
```

view_redundancy_group

Shows the present configuration of the redundancy group.

Format

```
emcli view_redundancy_group
      -redundancyGroupName="redGrpName"
```

Options

- **redundancyGroupName**

  You must specify a single redundancy group name. The target name should be the same as present in the repository, and it should be of target type= "generic_redundancy_group".

Examples

The following example shows the details for the 'redGrp1' Redundancy Group.

```
emcli view_redundancy_group -redundancyGroupName='redGrp1'
```

3

Error Code Reference

This chapter documents 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.
clear_credential	1—Target type does not exist. 2—Target does not exist. 3—Credential set does not exist. 4—Insufficient privileges. 5—Credential column does not exist.
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.
disable_audit	223—Syntax Error.

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_audit	223—Syntax Error.
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.
grant_privs	2—User does not exist.
	3—Invalid privilege.
	4—Invalid target privilege.
	5—Invalid globally unique identifier (GUID).
	6—One or more targets are not groups.
	7—Specified job does not exist.
	8—Privilege grant failed.
grant_roles	2—User does not exist.
	7—Role does not exist.
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.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
login	0—Verb success exit value. 1—Cannot establish an OMS connection storage area, or a corrupt area already exists. 2—A connection with the OMS cannot be established. 3—The login with the credentials provided failed at the OMS. 4— The Enterprise Manager license was not accepted by the current user. 5—The user is already logged in Enterprise Manager. 223—Command syntax error Verb exit value. 241—Custom attribute error handling. 255—Error code for browser-related errors.
logout	0—Verb success exit value. 1—Cannot establish an OMS connection storage area, or a corrupt area already exists. 2—A connection with the OMS cannot be established. 3—The login with the credentials provided failed at the OMS. 4— The Enterprise Manager license was not accepted by the current user. 249—OMS connection error verb exit value. 255—Error code for browser-related errors.
modify_aggregate_service	1—Target does not exist. 2—Target exists.
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.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
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>" .
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.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
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.
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.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
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

Verb	Error Code
relocate_targets	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: `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}; "` ■ Errors relocating targets from Source Agent to Destination Agent: < error message > < error message > ■ Exception in parsing targets from the command line argument <message>.
remove_beacon	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.
remove_service_system_assoc	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.
retry_job	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.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
revoke_roles	2—User does not exist.
	7—Role does not exist.
revoke_privs	2—User does not exist.
	3—Invalid privilege.
	4—One or more targets are invalid.
	5—Invalid globally unique identifier (GUID) privilege.
	6—One or more targets are not groups.
	7—Specified job does not exist.
	8—Privilege grant failed.
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.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
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.
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.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
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.
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_audit_settings	223—Syntax error, which could be an invalid directory name or invalid audit settings.

Table 3–4 (Cont.) Built-In Verb Errors

Verb	Error Code
update_db_password	1—Invalid target. 2—Invalid key value parameter. 3—Invalid old password. 4—Invalid privilege. 223—Syntax error.
update_host_password	1—Invalid target. 2—Invalid key value parameter. 3—Invalid old password. 4—Invalid privilege. 223—Syntax error.
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.