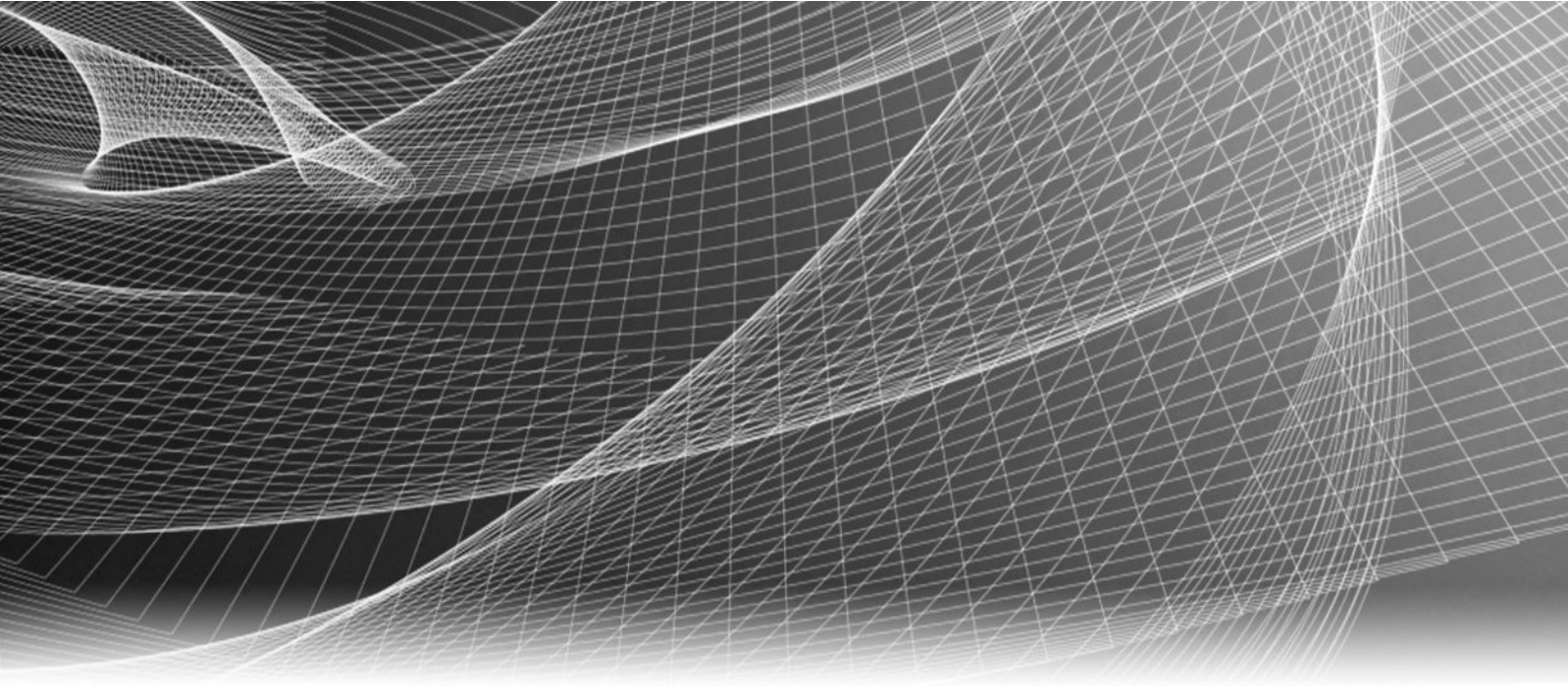

EMC® Avamar ® Management Console Command Line Interface (MCCLI)

Version 7.3

Programmer Guide

302-002-850

REV 01

EMC²

Published April, 2016

EMC Corporation
Hopkinton, Massachusetts 01748-9103
1-508-435-1000 In North America 1-866-464-7381
www.EMC.com

CONTENTS

TABLES

PREFACE

As part of an effort to improve its product lines, EMC periodically releases revisions of its software and hardware. Therefore, some functions described in this document might not be supported by all versions of the software or hardware currently in use. The product release notes provide the most up-to-date information on product features.

Contact your EMC technical support professional if a product does not function properly or does not function as described in this document.

Note

This document was accurate at publication time. Go to EMC Online Support (https://support.emc.com) to ensure that you are using the latest version of this document.

Purpose

This document describes how to install, configure, and use the Avamar Management Console Command Line Interface (MCCLI) Java client software application.

Audience

This document is intended for system administrators who are responsible for installing software and maintaining servers and clients on a network. This document assumes that the reader is familiar with the Avamar Administrator graphical management console as documented in the EMC Avamar Administration Guide, and does not generally repeat information in that document.

Revision history

The following table presents the revision history of this document.

Table 1 Revision history

Revision	Date	Description
01	April, 2016	GA release of Avamar 7.3

Related documentation

The following EMC publications provide additional information:

- *EMC Avamar Release Notes*
- *EMC Avamar Administration Guide*
- *EMC Avamar Operational Best Practices*
- *EMC Avamar and EMC Data Domain System Integration Guide*

Special notice conventions used in this document

EMC uses the following conventions for special notices:

NOTICE

Addresses practices not related to personal injury.

Note

Presents information that is important, but not hazard-related.

Typographical conventions

EMC uses the following type style conventions in this document:

Bold	Use for names of interface elements, such as names of windows, dialog boxes, buttons, fields, tab names, key names, and menu paths (what the user specifically selects or clicks)
Italic	Use for full titles of publications referenced in text
`Monospace`	Use for: • System code • System output, such as an error message or script • Pathnames, file names, prompts, and syntax • Commands and options
Monospace italic	Use for variables
`Monospace bold`	Use for user input
[ ]	Square brackets enclose optional values
|	Vertical bar indicates alternate selections - the bar means “or”
{ }	Braces enclose content that the user must specify, such as x or y or z
...	Ellipses indicate nonessential information omitted from the example

Where to get help

The Avamar support page provides access to licensing information, product documentation, advisories, and downloads, as well as how-to and troubleshooting information. This information may enable you to resolve a product issue before you contact EMC Customer Support.

To access the Avamar support page:

1. Go to https://support.EMC.com/products.
2. Type a product name in the **Find a Product** box.
3. Select the product from the list that appears.
4. Click the arrow next to the **Find a Product** box.
5. (Optional) Add the product to the **My Products** list by clicking **Add to my products** in the top right corner of the **Support by Product** page.

Documentation

The Avamar product documentation provides a comprehensive set of feature overview, operational task, and technical reference information. Review the following documents in addition to product administration and user guides:

- Release notes provide an overview of new features and known limitations for a release.
- Technical notes provide technical details about specific product features, including step-by-step tasks, where necessary.
- White papers provide an in-depth technical perspective of a product or products as applied to critical business issues or requirements.

Knowledgebase

The EMC Knowledgebase contains applicable solutions that you can search for either by solution number (for example, esgxxxxxx) or by keyword.

To search the EMC Knowledgebase:

1. Click the **Search** link at the top of the page.
2. Type either the solution number or keywords in the search box.
3. (Optional) Limit the search to specific products by typing a product name in the **Scope by product** box and then selecting the product from the list that appears.
4. Select **Knowledgebase** from the **Scope by resource** list.
5. (Optional) Specify advanced options by clicking **Advanced options** and specifying values in the available fields.
6. Click the search button.

Online communities

Visit EMC Community Network at http://community.EMC.com for peer contacts, conversations, and content on product support and solutions. Interactively engage online with customers, partners and certified professionals for all EMC products.

Live chat

To engage EMC Customer Support by using live interactive chat, click **Join Live Chat** on the **Service Center** panel of the Avamar support page.

Service Requests

For in-depth help from EMC Customer Support, submit a service request by clicking **Create Service Requests** on the **Service Center** panel of the Avamar support page.

Note

To open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

To review an open service request, click the **Service Center** link on the **Service Center** panel, and then click **View and manage service requests**.

Facilitating support

EMC recommends that you enable ConnectEMC and Email Home on all Avamar systems:

- ConnectEMC automatically generates service requests for high priority events.
- Email Home emails configuration, capacity, and general system information to EMC Customer Support.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Send your opinions of this document to DPAD.Doc.Feedback@emc.com.

Please include the following information:

- Product name and version
- Document name, part number, and revision (for example, 01)
- Page numbers

- Other details that will help us address the documentation issue

CHAPTER 1

Introduction

This chapter includes the following topics:

System requirements

To connect to the Avamar server using MCCLI, you must have:

- A valid Avamar Administrator ID and password
- Network access to an operational Avamar server

Additionally, the computer running MCCLI software must meet the minimum requirements listed in the following table.

Table 2 Minimum system requirements for MCCLI software

Requirement	Minimum
Operating system	Red Hat Enterprise Linux Release 4 (64-bit) SUSE Linux Enterprise Server 11 (64-bit)
RAM	256 MB
Hard drive space	60 MB permanent hard drive space
Network interface	10BaseT or higher, configured with the latest drivers for the platform
Java Runtime Environment	JRE 1.7

Capabilities and limitations

These are the known capabilities and limitations of the MCCLI.

Version compatibility
The MCCLI software must be the same version as the MCS.

Hierarchical management
Hierarchical management is not supported. The root domain is assumed for all user IDs included on the command line. This means that a user must have an account in the root domain to use the MCCLI, and domain administrators cannot log on.

Relative path filenames
The use of filenames containing relative paths is not supported. Filenames are specified as part of the client bulk validate and load commands.

Client bulk validate/load file format
The `client import-clients-from-file` and `client validate-clients-from-file` commands currently only support clients definition input files in XML format. You cannot use a CSV file with these commands.

Delayed updates for MCCLI scheduler changes
When you use the MCCLI to suspend or resume the scheduler, the change takes effect immediately. However, the change may not appear in Avamar Administrator for approximately 15 seconds.

Deleting clients with backups pending
As with the Avamar Administrator, you cannot delete a client until it is idle without any backups in progress or it is in the wait queue.

Performance

MCCLI commands typically require approximately 8 seconds to complete. This elapsed time is primarily because the Java Virtual Machine (JVM) must be started to run any MCCLI command, and because decryption framework initialization adds approximately 6 seconds.

If you provide the option directly with each command, you can bypass the decryption framework initialization, which will improve performance.

Maximum number of clients

As with Avamar Administrator, the MCCLI is suitable for systems with a maximum of 5,000 clients.

Java stack trace

Any `mccli` command might display a Java stack trace if the command fails. This is due to unhandled exceptions.

Generic mccli command failed message

Some errors and failure conditions are not associated with specific event codes. Therefore, when these errors and failure conditions occur, the generic event code for a failed `mccli` command appears. For example, missing arguments or failed connection from the MCCLI to the MCS generates the same `23998` event (`mccli command failed`).

Remote clients must run mccli commands as root

If you have installed MCCLI on a remote client, all `mccli` commands invoked using the operating system root privileges.

Furthermore, if additional security is required, users should remove any passwords stored in `mcclimcs.xml` and instead supply an `--mcspasswd` option with each `mccli` command.

Requirements for accounts running MCCLI scripts

The following requirements apply to scripting with the MCCLI:

- MCCLI scripts must be run using a local service-type account.
- Use of the MCUser account is not supported.
- The service account should only be used for scripts, and not as a regular user.

The EMC Avamar Administration Guide contains information about creating a new Avamar server account. After creating a new account, you must add the account profile to the mcclimcs.xml file as show in Add a new service account profile to the mcclimcs.xml file and encrypt the account password on page 140 .

CHAPTER 2

Installation and Configuration

This chapter includes the following topics:

Installing the Java Runtime Environment

The MCCLI requires Java Runtime Environment (JRE) 1.6 Update 12 or later. You can download a current JRE directly from the Avamar server.

Procedure

1. On the computer where MCCLI will be installed, open a web browser and type the following URL:

   http://*Avamar-server*

   where *Avamar-server* is the Avamar server network hostname or IP address.

   The EMC Avamar Web Restore web page appears.
2. Click **Downloads**.
3. Expand the **Linux for x86 (64 bit)** folder.
4. Expand the correct Linux operating system folder for your computer.
5. Locate the JRE RPM install package (it is typically the last entry in the folder).
6. If the JRE on the client computer is older than the JRE hosted on the Avamar server, download the `jre-version-platform.rpm` install package to a temporary install folder such as `/tmp`.

   where:

   - *version* is the JRE version.
   - *platform* is the computing platform.
7. Open a command shell and log in as root.
8. Change directory to the temporary install folder. For example:

   cd /tmp
9. Follow the onscreen instructions to complete the JRE installation.

Installing and configuring the MCCLI software

Before you begin

The MCCLI sotware requires Java Runtime Environment (JRE) 1.6 Update 12 or later.

Procedure

1. On the computer where MCCLI will be installed, open a web browser and type the following URL:

   http://*Avamar-server*

   where *Avamar-server* is the Avamar server network hostname or IP address.

   The EMC Avamar Web Restore web page appears.
2. Click **Downloads**.
3. Expand the **Linux for x86 (64 bit)** folder.
4. Do one of the following:

   - If installing on SUSE Linux Enterprise Server 11, expand the **SUSE Linux Enterprise Server 11** folder.

- If installing on Red Hat Enterprise Linux Release 4, expand the **Red Hat Enterprise Linux 4** folder.

5. Download the `dpnmccli-version.platform.x86_64.rpm` install package to a temporary install folder such as `/tmp`.

   where:

   - *version* is the MCCLI software.
   - *platform* is either `sles11_64` or `rhel4_64`.

6. Open a command shell and log in as root.
7. Change directory to the temporary install folder. For example:

   `cd /tmp`

8. Install the MCCLI RPM by typing:

   `rpm -ivh dpnmccli-version.platform.x86_64.rpm`

   Note

   If installing the MCCLI software on the same Linux client that already has Avamar Administrator installed, you must include the `--force` option.

9. Follow the onscreen instructions to complete the MCCLI software installation.
10. Configure the MCCLI software by typing `avsetup_mccli`.
11. When prompted, type the following information, and then press **Enter** to complete each entry.
    - Full path to the JRE installation folder. The default location is `/usr/java/jre1.7.0_72` or `/usr/java/jre1.8`.
    - Full path to Avamar software installation folder. The default location is `/usr/local/avamar`.
    - Full path to the folder where user data is stored. The default location is `~/.avamardata/var`.
    - Avamar server IP address or hostname as defined in corporate DNS.
    - MCS data port number. The default data port is 7778.
    - Avamar administrative user account name.
    - Avamar administrative user account password.

CHAPTER 3

Command Reference

This chapter includes the following topics:

General programming notes

The `mccli` is a shell script wrapper that invokes the MCCLI Java application. It automatically sets various environment arguments that are required to invoke a Java application, thereby simplifying use of the MCCLI java application.

Data types

The following table lists the data types used by `mccli` commands and discusses how to correctly specify each data type on the `mccli` command line.

Table 3 Data types for `mccli` commands

Data type	Description
Boolean	*Boolean* values are the case-insensitive words `true` or `false`.
Integer	*Integer* values are always whole numbers.
String	*String* values contain plain text. If a string value contains spaces, it must be enclosed in either single or double quotes. Some `mccli` *String* values accept regular expression (regex) pattern matching operators, also known as wildcards.

Default values

If a command line argument has a default value, it is shown in parentheses. For example, the following *Boolean* value defaults to the false condition:

`--verbose=`*`Boolean(false)`*

Similarly, the following string value defaults to the root domain denoted by the slash character (/):

`--domain=`*`String(/)`*

Optional and required arguments

Command synopses use the following convention to convey whether a particular argument is required for that command:

- Optional arguments are enclosed by square brackets
- Required arguments are not

For example, consider the following example:

`[--domain=`*`String(/)`*`] --name=`*`String`*

The presence of square brackets indicates that `--domain=` is an optional argument, while the absence of square brackets indicates that `--name=` is a required argument.

Pattern matching

Some `mccli` string values accept regular expression (regex) pattern matching operators, also known as wildcards. `mccli` pattern matching operators are subject to certain capabilities and limitations described in the following table.

Table 4 Pattern matching operator capabilities and limitations

Operator	Description
Asterisk (*)	Matches zero or more occurrences of any character until the first folder delimiter character (for example, slash on UNIX platforms and backslash on Windows platforms) is encountered. This effectively limits the pattern matching to a single client folder. For example, `/usr/*` matches the contents of `/usr` but not the contents of `/usr/bin` or `/usr/local`.
Double asterisk (**)	Matches zero or more occurrences of any character. This correlates to conventional single asterisk regex behavior. For example, `/usr/**` matches the entire `/usr` folder structure, no matter how many subfolder levels are encountered.
Question mark (?)	Matches one occurrence of any character. Conventional regex behavior.
Plus sign (+)	Unlike conventional regex, the plus sign is not processed as a glob operator; the plus sign only matches a single occurrence of the plus sign.
Forward slash (/)	Patterns beginning with forward slash (/) are assumed to be absolute path designations for a single folder. Recursive processing of subfolders is disabled, and that folder name is not matched anywhere else.
[range of values]	Characters enclosed in square brackets and separated by a single hyphen (-) are interpreted as a range of values. This is conventional regex behavior. For example: • `[0-9]` matches any single numeric character • `[a-z]` matches any single lowercase alpha character
Pound sign (#)	In most cases, you can define multiple matching patterns in a text file and pass that file into the utility. This is generally easier than specifying multiple matches directly on the command line. However, when using a text file to pass in pattern matches, the pound sign (#) is interpreted as a comment if it appears at the beginning of a matching pattern. This causes that entire pattern matching entry to be ignored.

Output description

Specific messages are returned when an `mccli` command completes either successfully or unsuccessfully.

When an `mccli` command completes successfully, the following message is returned:

```
0,23000,CLI command completed successfully.
```

The message comprises three separate comma-delimited elements:

- The first element is the numeric return code, which is zero (`0`) because the command successfully completed.
- The second element is the numeric event code, `23000`.
- The third element is the event code short description, `CLI command completed successfully`.

When an `mccli` command does not complete successfully, the output message is in the same format, but the return code is one (`1`) and the event code describes the error condition. For example:

```
1,22288,Dataset does not exist.
```

Command line syntax

This is the proper format and syntax for all `mccli` commands.

Syntax

```
mccli resource-class command command-options
global-options display-options
```

Resource classes

Each `mccli` command line must specify one and only one of the resource classes.

`activity`
: Cancel or show backup, restore, or validation activities.

`agent`
: Shows summary properties for all client agents.

`backup`
: Restore folders and files to backup clients, and manage backups stored on the Avamar server.

`checkpoint`
: Manage Avamar server checkpoints.

`client`
: Manage backup client accounts on the Avamar server.

`dataset`
: Manage backup datasets on the Avamar server.

`dd`
: Manage Data Domain systems for use as Avamar backup targets.

`domain`
: Manage Avamar server domains and subdomains.

`dump`
: Dumps various DPNProxyService caches for troubleshooting purposes. Strictly reserved for EMC internal use only.

`event`
: Access and manage Avamar server event code information.

`group`
: Manage groups and group policy.

`help`
: Show online help on a resource-by-resource basis.

`mcs`
: Control various Management Console Server (MCS) functions.

`plugin`
: Manage client plug-ins on the Avamar server.

`retention`
: Manage backup retention policies.

`schedule`
: Manage Avamar server schedules.

`server`
: Monitor various Avamar server functions.

`user`
: Manage backup user accounts.

`vcenter`

Browses a vCenter to locate virtual machines.

`version`

Shows the version of `mccli` currently installed.

`vmcache`

Assists with debugging possible vCenter data cache synchronization issues. Strictly reserved for EMC internal use only.

Commands

The list of possible commands is specific to each resource class. Individual resource class listings provide a list of commands available for each resource class.

Command options

The list of possible command options is specific to each combination of resource class and command. The individual command listings provide a list of command options available for each command.

Global options

Each `mccli` command line can contain one or more of the following options, which are global in nature, meaning that they can be used with any resource class or command.

`--mcsprofile=`*String*

Specifies the MCS profile name. If you supply this option, then settings stored in this profile are used, and other global options that you supply on the command line are ignored.

`--mcsaddr`*String*

Specifies the MCS network name or IP address.

`--mcsuserid`*String*

`--mcspasswd`*String*

Specifies an Avamar user account and password that is used to run `mccli` commands.

`--mcsport`*Integer*

Specifies the data port used to contact the MCS.

Typically, global options are persistently stored and read from the `mccli.xml` preferences file. You only supply the global options on an `mccli` command line in cases where the persistent settings must be temporarily overridden.

Display options

Each `mccli` command line can contain either of the following options, which control how command output is displayed.

`--xml`

Formats output as XML. This is useful for parsing output as part of a script.

`--normalize`

Produces output in a format that better supports parsing and comparison. This option affects any command that returns a date or time, or file system information such as capacity or file sizes.

The following table shows examples of returned values expressed in both default and normalized formats.

Table 5 Returned values in default and normalized formats

Value	Default format	Normalized format
Absolute time	Expressed as a conventional date and timestamp with the local timezone. For example: `2011-09-12 11:00:17 PDT`	Expressed as an integer representing the UNIX precision time format numerical (UTC

Table 5 Returned values in default and normalized formats (continued)

Value	Default format	Normalized format
		milliseconds from the epoch). For example: `1189620017000`
Elapsed time	Expressed as the number of days, hours, minutes, and seconds that have elapsed. For example: `12 days 17h:30m`	Expressed as an integer representing the total number of milliseconds that have elapsed. For example: `1099814000`
Capacity	Expressed in bytes, MB, GB, or TB, whichever is most correct. For example: `1.3 TB`	Expressed as an integer representing the total number of bytes. For example: `1464957140992`

Enumerating valid object IDs

Several `mccli` command options require precise case-sensitive string or numeric object IDs for input values.

The following table lists `mccli` commands that return lists of valid object IDs.

Table 6 Enumerating valid object IDs

Object ID	Data type	Command
Activity ID	String	`mccli activity show`
Backup label	Integer	`mccli backup show`
Checkpoint ID	String	`mccli checkpoint show`
Client name	String	`mccli client show`
Dataset name	String	`mccli dataset show`
Domain name	String	`mccli domain show`
Event ID	Integer	`mccli event show`
Group name	String	`mccli group show`
Plug-in ID	Integer	`mccli plugin show`
Retention policy name	String	`mccli retention show`
Schedule name	String	`mccli schedule show`
Time zone name	String	`mccli schedule show-timezones`
Service name	String	`mccli server show-services`
User account names	String	`mccli user show`

Table 6 Enumerating valid object IDs (continued)

Object ID	Data type	Command
Authentication system name	String	`mccli user show-auth`

Global event codes

These events codes are global, which means that they can be returned for any `mccli` command. However, in the event of an error, many `mccli` commands return other event codes that more fully describe the specific error condition encountered. Those event codes are listed with each `mccli` command.

22601	Server inactive.
23000	CLI command successfully completed.
23001	Arguments required by CLI command are either missing or empty.
23993	Attempt to read or write a file has failed.
23995	Invalid option specified on the CLI.
23996	Failed to connect to the administrator server.
23997	Conflicting arguments specified on the command line.
23998	CLI command failed.
23999	Unexpected command failure.

Deprecated resources and commands

Deprecated resources and commands will continue to work, but should not be used for new integrations. If documentation for these deprecated resources and commands is required, refer to previous versions of this publication.

The following tables list deprecated resources and commands.

Table 7 Deprecated resources

Deprecated resource	Use instead
`snapup`	`backup`

Table 8 Deprecated commands

Deprecated command	Use instead
`client activate`	`client invite`
`client load-bulk`	`client import-clients-from-file`
`client snapup-dataset`	`client backup-dataset`
`client snapup-group-dataset`	`client backup-group-dataset`
`client snapup-target`	`client backup-target`

Table 8 Deprecated commands (continued)

Deprecated command	Use instead
`client validate-bulk`	`client validate-clients-from-file`
`group show-client-members`	`group show-members`
`group snapup`	`group backup`

activity

The `mccli activity` resource is used to cancel or show backup, restore, and validation activities.

activity cancel

The `mccli activity cancel` command cancels a backup, restore, or validation activity. If the activity has already completed, then the command returns an event code indicating that the activity ID is invalid, along with information indicating that the job has already completed.

Syntax

```
mccli activity cancel --id=String --wait[=min]
```

Options

`--id=`*String*
: Cancels the specified activity ID. *String* must be a valid activity ID. This argument is required.
Use `mccli activity show` to return a list of valid activity IDs.

`--wait`[=*min*]
: Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22205	Backup cancelled via console.
23010	Invalid ID specified on the CLI.
23023	Activity already completed.

activity get-log

The `mccli activity get-log` command displays the contents of the session log file for an activity. Use the `--xml` display option to show escaped log file content.

Syntax

```
mccli activity get-log --id=String
```

Options

`--id=`*String*

Gets log files for this activity ID. *String* must be a valid activity ID. This argument is required.

Use `mccli activity show` to return a list of valid activity IDs.

Event codes

23010	Invalid ID specified on the CLI.

activity show

The `mccli activity show` command lists backup, restore, and validation activities with summary information, or detailed information for a specific activity. If you are viewing summary information for multiple activities, you can filter the information on a domain or client basis.

Syntax

```
mccli activity show
{--domain=String(/) |  --name=String [--domain=String(/)]}
[--active=Boolean(false)] [--completed=Boolean(false)]
[--contained-vm-activities=Boolean(false)] [--id=String]
[--name=String] [--queued=Boolean(false)]
[--source={avamar | dd}] [--verbose=Boolean(false)]
```

Options

`--active=`*Boolean(false)*

If `true`, then only currently running activities are shown.

`--completed=`*Boolean(false)*

If `true`, then only completed activities are shown.

`--contained-vm-activities=`*Boolean(false)*

If true, shows activities for virtual machine clients within VMware containers or vApps. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--domain=`*String(/)*

If supplied without `--name`, shows all activities for that domain.

If supplied with `--name`, specifies the Avamar server domain where that client resides.

`--id=`*String*

Specifies which activities to show. *String* must be a valid activity ID. Multiple `--id` options can be specified on the same command line.

Use `mccli activity show` to return a list of valid activity IDs.

`--name=`*String*

Shows activities for the specified client.

If *String* is a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` option is ignored.

Use `mccli client show` to return a list of valid client names.

`--queued=`*Boolean(false)*

If `true`, then only queued activities are shown.

`--source={avamar | dd}`

If `avamar` is specified, then the command shows activities for the Avamar server.

If`dd` is specified, then the command shows activities for all configured Data Domain systems.

`--verbose=`*Boolean(false)*

If `true`, then detailed activity information is returned.

If `false` or not supplied, then summary information is returned.

Examples

This command returns activities for all virtual machine clients in the 10.31.183.55/FO2 container, with output formatted as XML:

```
mccli activity show --name=/10.31.183.55/FO2
--contained-vm-activities=true --verbose=true --xml
```

```
<CLIOutput>
  <Results>
   <ReturnCode>0</ReturnCode>
   <EventCode>23000</EventCode>
   <EventSummary>CLI command completed successfully.</EventSummary>
  </Results>
  <Data>
   <Row>
    <ID>9134224603666509</ID>
    <Status>Completed</Status>
    <ErrorCode>0</ErrorCode>
    <StartTime>2013-07-14 06:07 UTC</StartTime>
    <Elapsed>00h:00m:46s</Elapsed>
    <EndTime>2013-07-14 06:08 UTC</EndTime>
    <Type>On-Demand Backup</Type>
    <ProgressBytes>4,200,757</ProgressBytes>
    <NewBytes>0.1%</NewBytes>
    <Client>TEST1</Client>
    <Domain>/10.31.183.55/HleDynamicClients</Domain>
    <OS>windows7Server64Guest</OS>
    <ClientRelease>7.1.100-333</ClientRelease>
    <Sched.StartTime>2013-07-14 06:07 UTC</Sched.StartTime>
    <Sched.EndTime>2013-07-15 06:07 UTC</Sched.EndTime>
    <ElapsedWait>00h:00m:29s</ElapsedWait>
    <Group>Admin On-Demand Group</Group>
    <Plug-In>Windows VMware Image</Plug-In>
    <RetentionPolicy>Default Retention</RetentionPolicy>
    <Retention>N</Retention>
    <Schedule>Admin On-Demand Schedule</Schedule>
    <Dataset>/Client On-Demand Data</Dataset>
    <WID>MOD-1342246036652_42992503ce8f11c007c26d73711fac08f5d5e331
    </WID>
    <Server>Avamar</Server>
    <Container>FO2</Container>
   </Row>
  </Data>
</CLIOutput>
```

agent show

The `mccli agent show` command shows summary properties for all client agents.

Syntax

```
mccli agent show
```

backup

The `mccli backup` resource is restore folders and files to backup clients, and manage backups stored on the Avamar server.

backup delete

The `mccli backup delete` command permanently deletes a backup from the server.

Syntax

```
mccli backup delete --name=String --created=String
--labelNum=Integer [--contained-vm-name=String]
[--domain=String(/)] [--force=Boolean(false)] [--location=String] [--
recursive=Boolean(false)]
```

Options

`--contained-vm-name=`*String*
: Specifies a virtual machine client within a VMware container or vApp. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--created=`*String*
: Specifies the date the backup was created. *String* must be in the format of YYYY-MM-DD. This argument is required.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.

`--force=`*Boolean(false)*
: By default, if you attempt to delete a backup that has more than one retention type assigned to it, then a warning is issued and the backup is not deleted. This is intended to prevent inadvertent deletion of a single backup that could remove more than one level of historical backups (daily, weekly, monthly, or yearly) from the server.
  If `true`, then the checking is disabled and the backup is deleted regardless of the number of retention types assigned to it.

`--labelNum=`*Integer*
: Specifies the label number of the backup to delete. This argument is required.
  Use `backup show` without supplying `--labelNum` to return a list of backups with integer label numbers.

`--name=`*String*
: Specifies the client from which the backup was originally taken. This argument is required.
  If you supply a fully qualified client name (for example, `/clients/MyClient`), then `--domain` is ignored.
  Use `mccli client show` to return a list of valid client names.

`--location=`*String*
: Specifies the location that backups were replicated to.

`--recursive=`*Boolean(false)*
: If true, recursively deletes the backups of child VMs.

Event codes

22236	Client does not exist.
22552	Backup does not exist.

22553	Backup deleted.
22556	Changed backup retention.
22558	Multiple retention tags exist.

Examples

This command deletes the first backup (`--labelNum=1`) created on 2014-07-14 for virtual machine client TEST1 in the 10.31.183.55/FO2 container:

`mccli backup delete --name=/10.31.183.55/FO2 --contained-vm-name=TEST1 --created="2014-07-14" --labelNum=1`

```
0,22553,Backup deleted.
Attribute   Value
----------  ------------------------------------------------------
labelnum    1
createtime  2014-07-14 06:08:21 UTC
path        /10.31.183.55/HleDynamicClients/UCKZ3FGLz70WRRu3nq2ZuA
retention   N
plugin      3016
```

backup edit

The `mccli backup edit` command enables you to change the backup expiration date.

You can change the backup expiration date by any of the following methods:

- Directly specifying a new expiration date, or that the backup should never expire
- Extending the existing expiration date
- Assigning an extended retention type to a backup

Syntax

```
mccli backup edit --name=String
--created=String --labelNum=Integer
{--expiration={YYYY-MM-DD | NO_EXPIRATION}
| --extend-expiration=+nn{D | W | M | Y}
| --retention={{D | daily} | {W | weekly}
| {M | monthly} | {Y | yearly} | none}}
[--contained-vm-name=String] [--domain=String(/)]
[--force=Boolean(false)] [--location=String]
```

Options

`--contained-vm-name=`*String*

Specifies a virtual machine client within a VMware container or vApp. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--created=`*String*

Specifies the date the backup was created. *String* must be in the format of YYYY-MM-DD. This argument is required.

`--domain=`*String(/)*

Specifies the Avamar server domain that contains the client specified by `--name`. Use `mccli domain show` to return a list of valid domain names.

`--expiration={`*YYYY-MM-DD*`|NO_EXPIRATION}`

Specifies a new expiration date.

- A specific calendar date, in the format of YYYY-MM-DD
- `NO_EXPIRATION` specifies that the backup should never expire

The `--expiration`, `--extend-expiration`, and `--retention` options are mutually exclusive.

`--extend-expiration=+`*nn*{`D` | `W` | `M` | `Y`}

Extends the existing expiration date. Where *nn* is an integer, valid values are:

- +*nn*`D`—number of additional days added to the existing backup expiration date
- +*nn*`W`—number of additional weeks added to the existing backup expiration date
- +*nn*`M`—number of additional months added to the existing backup expiration date
- +*nn*`Y`—number of additional years added to the existing backup expiration date

The `--expiration`, `--extend-expiration`, and `--retention` options are mutually exclusive.

`--force=`*Boolean(false)*

By default, if you attempt to delete a backup that has more than one retention type assigned to it, then a warning is issued and the backup is not deleted. This is intended to prevent inadvertent deletion of a single backup that could remove more than one level of historical backups (daily, weekly, monthly, or yearly) from the server.

If `true`, then the checking is disabled and the backup is deleted regardless of the number of retention types assigned to it.

`--labelNum=`*Integer*

Specifies the label number of the backup to delete. This argument is required.

Use `backup show` without supplying `--labelNum` to return a list of backups with integer label numbers.

`--location=`*String*

Specifies the location that backups were replicated to.

`--name=`*String*

Specifies the client from which the backup was originally taken. This argument is required.

If you supply a fully qualified client name (for example, `/clients/MyClient`), then `--domain` is ignored.

Use `mccli client show` to return a list of valid client names.

`--retention=`{{`D` | `daily`} | {`W` | `weekly`} | {`M` | `monthly`} | {`Y` | `yearly`} | `none`}

Specifies the extended retention types to assign to this backup.

Both short form and long form`none` retention type values are allowed and can be mixed. For example, all of the following are valid:

- `--retention=D,weekly`
- `--retention=Daily,W,monthly`

The `--expiration`, `--extend-expiration`, and `--retention` options are mutually exclusive.

Event codes

22236	Client does not exist.
22552	Backup does not exist.
22556	Changed backup retention.
22557	Failed to modify retention of a backup.
22558	Multiple retention tags exist.

Examples

This command changes backup retention to daily for the second backup (`--labelNum=2`) created on 2014-07-14 for virtual machine client TEST1 in the 10.31.183.55/FO2 container:

```
mccli backup edit --name=/10.31.183.55/FO2
--contained-vm-name=TEST1 --labelNum=2 --created=2014-07-14
--retention=D
```

```
0,22556,Changed backup retention.
Attribute   Value
----------  ------------------------------------------------------
labelnum    2
createtime  2014-07-14 06:08:21 UTC
path        /10.31.183.55/HleDynamicClients/UCKZ3FGLz70WRRu3nq2ZuA
retention   D
plugin      3016
```

backup restore

The `mccli backup restore` command restores data to a client.

This command returns an activity ID, which can be passed to `mccli activity show` to get status for the restore activity.

This command supports any of the following restore scenarios:

- If restoring from a normal (non-virtual) filesystem or application backup:
  - Restore entire backup to the same (non-virtual) client
  - Restore selected folders or files to the same (non-virtual) client
  - Redirected restore of selected folders or files to a different (non-virtual) client
- If restoring from a VMware image backup:
  - Restore an entire VMware image backup to the same virtual machine
  - Redirected restore an entire VMware image backup to a different existing virtual machine
  - Redirected restore an entire VMware image backup to the new virtual machine
  - Restore selected folders or files to the same virtual machine
  - Redirected restore of selected folders or files to a different virtual machine

Syntax

```
mccli backup restore --name=String --labelNum=Integer
[--cmd=String [--cmd=String ...]]
[--contained-vm-name=String] [--data={all | vmdk-filename}]
[--datacenter=String] [--datastore-name=String]
[--dest-client-domain=String] [--dest-client-name=String]
[--dest-client-username=String --dest-client-password=String]
[--dest-dir=String] [--domain=String(/)]
[--esx-host-name=String] [--esx-host=String] [--folder=String] [--
location=String]
[--network-old=String --network-new=String]
[--plugin=Integer] [--resource-pool=String]
[--restore-vm-to={original | existing | new | flr}
[--virtual-center-name=String] [--vm-configuration=Boolean(false)]
[--wait[=min]]
```

Options

`--cmd=`*String*

Specifies one ore more optional plug-in commands. You can supply multiple `--cmd` arguments, but each argument can only specify one plug-in command.
For example, this is valid `--cmd` syntax:
`--cmd="verbose=5" --cmd="throttle=5"`
However, this is not valid `--cmd` syntax:
`--cmd="verbose=5 throttle=5"`

`--contained-vm-name=`*String*
Specifies a virtual machine client within a VMware container or vApp. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--data=`{`all` | *vmdk-filename*}
Specifies a single optional target folder or file to restore.
When restoring from a VMWare image backup:

- `all`—restores all virtual disks
- *vmdk-filename*—restores only the virtual disk defined by *vmdk-filename*

Each `--data=` option can only specify one target folder or file to restore. Use multiple `--data` options to restore multiple targets.

`--datacenter=`*String*
When restoring a virtual machine or vApp, specifies a fully qualified datacenter name in vCenter.
This argument is required when restoring an entire image or selected disks to the same, different existing, or new virtual machine. It is not valid when restoring individual folders or files.

`--datastore-name=`*String*
When restoring a virtual machine or vApp to a new virtual machine, specifies the datastore name to be removed from the Avamar proxy. This argument is required.

`--dest-client-domain=`*String*
Specifies the Avamar server domain that contains the alternative client specified by the `--dest-client-name` argument.
Use `mccli domain show` to return a list of valid domain names.

`--dest-client-name=`*String*
When performing any redirected restore, specifies the destination client.
When performing a redirected restore of specific folders or files, you must also supply `--dest-dir`.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--dest-client-domain` option is ignored.
Use `mccli client show` to return a list of valid client names.

`--dest-client-username=`*String*

`--dest-client-password=`*String*
When restoring folders or files from a VMware image backup, specifies the destination client username and password.

`--dest-dir=`*String*
When performing a redirected restore of specific folders or files, specifies the destination folder for the restored folders or files.
This argument is required when performing any redirected restore of specific folders or files, and optional for all other operations.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the client specified by `--name`.
Use `mccli domain show` to return a list of valid domain names.

`--esx-host-name=`*String*
When restoring a VMware image backup to a new virtual machine, specifies a fully qualified ESX server hostname in a datacenter.

`--esx-host=`*String*

When restoring a VMware image backup to a new a vApp, specifies the path to a host or cluster inside the datacenter.

`--folder=`*String*

When restoring a virtual machine or vApp to an existing or new virtual machine, specifies the folder path for the destination virtual machine in the datacenter.

`--labelNum=`*Integer*

Specifies the label number of the backup to delete. This argument is required.

Use `backup show` without supplying `--labelNum` to return a list of backups with integer label numbers.

`--location=`*String*

Specifies the location that backups were replicated to.

`--name=`*String*

Specifies the client from which the backup was originally taken. This argument is required.

If you supply a fully qualified client name (for example, `/clients/MyClient`), then `--domain` is ignored.

Use `mccli client show` to return a list of valid client names.

`--network-old=`*String*

`--network-new=`*String*

When restoring a VMware image backup to a new a vApp, these options map old network setting to a new network setting. Both `--network-old` and `--network-new` options must be supplied.

Multiple `--network-old` and `--network-new` pairs are allowed. Multiple comma-separated mapping values are allowed for each option.

Example one-to-one mapping:

```
--network-old=vlan1 --network-new=vlan241
```

Example multiple one-to-one mappings:

```
--network-old=vlan1 --network-new=vlan241
--network-old=vlan2 --network-new=vlan242
--network-old=vlan3 --network-new=vlan243
```

Example many-to-one mapping:

```
--network-old=vlan1,vlan2,vlan3 --network-new=vlan241
```

Example combined many-to-one with one-to-one mapping:

```
--network-old=vlan1,vlan2 --network-new=vlan241
--network-old=vlan3 --network-new=vlan242
```

`--plugin=`*Integer*

Specifies the plug-in ID for the restore. This argument is required for all restores except vApps.

When performing a redirected restore, this argument should specify a plug-in that is compatible with the alternative destination client, not the plug-in that was originally used to perform the backup.

Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

When restoring vApps, multiple `--plugin` and `--cmd` arguments are allowed. For example:

```
--plugin=3016 --cmd=--test=true --cmd=--test1=false
--plugin=1016 --cmd=--test3=true
```

In this example, `--test` and `--test1` are processed by plug-in 3016; `--test3` is processed by plug-in 1016.

`--resource-pool=`*String*
When restoring a VMware image backup to a new a vApp, specifies a resource pool on the ESX host.

`--restore-vm-to={original|existing|new|flr}`
When restoring a VMware image backup, specifies the type of restore operation. String must be one of the following:

- `original`—restores the entire backup image to the original virtual machine or vApp
- `existing`—restores the entire backup image to a different existing virtual machine
- `new`—restores the entire backup image to a new virtual machine or vApp
- `flr`—restores folders or files

`--virtual-center-name=`*String*
When restoring a VMware image backup to an existing or new virtual machine or vApp, specifies the vCenter name.

`--vm-configuration=`*Boolean(false)*
When restoring a VMware image backup to an existing virtual machine or vApp, and this argument is `true`, VMware configuration files are restored.

`--wait`[=*min*]
Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22236	Client does not exist.
22297	Request to restore is rejected.
22312	Client restore scheduled.
23009	Invalid plugin specified on the CLI.

Examples

This command restores an entire VMware image backup to the original virtual machine:

```
mccli backup restore --name=/Test/Test1 --labelNum=1 --plugin=3016
--restore-vm-to=original
```

```
0,22312,client restore scheduled.
Attribute    Value
-----------  ----------------------------------------------
client       /Test/Test1
activity-id  9131666434371809
```

This command restores an entire VMware image backup to a different existing virtual machine:

```
mccli backup restore --name=/Test/Test1 --labelNum=1 --plugin=3016
--restore-vm-to=existing --datacenter=Test/ESX40 --virtual-center=Test
--dest-client-name=abcdef
```

```
0,22312,client restore scheduled.
Attribute    Value
-----------  ----------------------------------------------
client       /Test/Test1
activity-id  9131666434371810
```

This command restores virtual disk from a VMware image backup to a new virtual machine:

```
mccli backup restore --name=/Test/Test1/ --labelNum=2 --plugin=3016
--restore-vm-to=new --datacenter=VAAYU-DEV-WIN
--virtual-center-name=vcenter-1.example.com --datastore-name=Storage1
--esx-host-name=10.31.183.7 --dest-client-name=Test12345
--folder=Discovered virtual machine
--data=[Storage1] Disk-1/Disk-1.vmdk
```

```
0,22312,client restore scheduled.
Attribute    Value
-----------  ----------------------------------------------
client       /Test/Test1/Disk-1_UohhwqYknag96PWb1WrEnA
activity-id  9131666434371811
```

This command restores selected folders and files to a different existing virtual machine:

```
mccli backup restore --name=/Test/Test1 --labelNum=1 --plugin=3019
--restore-vm-to=flr --dest-client-name=Test5
--dest-client-domain=/Test/Test --dest-client-username=Administrator
--data=C:\\1.log --dest-dir=C:\\ --dest-client-password=abcdefgh
```

```
0,22312,client restore scheduled.
Attribute    Value
-----------  ----------------------------------------------
client       /Test/Test1/Test5
activity-id  9131666434371812
```

This command restores a vApp to the original vApp:

```
mccli backup restore --name=/10.31.183.55/ABCD1
--contained-vm-name=YYY --labelNum=1
--restore-vm-to=original --vm-configuration=true
```

```
0,22312,client restore scheduled.
Attribute    Value
-----------  ----------------------------------------------
client       /10.31.183.55/HleDynamicClients/YYY_EDEYNV5CM
activity-id  9131666434371813
```

This command restores a vApp to the original vApp with plug-in command line options:

```
mccli backup restore --name=/10.31.183.55/ABCD1
--contained-vm-name=YYY --labelNum=1
--restore-vm-to=original --plugin=3016
--cmd=--allnodes=false --cmd=--test=true
--plugin=1016 --cmd=--allnodes=true
```

```
0,22312,client restore scheduled.
Attribute    Value
-----------  ----------------------------------------------
client       /10.31.183.55/HleDynamicClients/YYY_EDEYNV5CM
activity-id  9131666434371814
```

This command restores a vApp to a new vApp:

```
mccli backup restore --name=/10.31.183.55/ABCD1
--contained-vm-name=YYY --labelNum=1
--restore-vm-to=new --datacenter=DCF1/DCF2
--virtual-center-name=10.31.183.55 --datastore-name=datastore1
--esx-host=HOF1/HOF2/10.31.183.17 --dest-clientname=MyRestorevApp
```

--folder=FO1/FO2 --resource-pool=RP1/RP2
--network-old=VM Network --network-new=VM Network

```
0,22312,client restore scheduled.
Attribute    Value
-----------  ----------------------------------------------
client       /10.31.183.55/HleDynamicClients/YYY_EDEYNV5CM
activity-id  9131666434371815
```

backup show

The `mccli backup show` command returns all backups currently stored on the Avamar server or a Data Domain system for a client.

Syntax

```
mccli backup show --name=String --labelNum=Integer
[--after=String] [--before=String]
[--contained-vm-name=String] [--dir=String]
[--domain=String(/)] [--recursive=Boolean(false)]
[--retention={{D | daily} | {W | weekly}
| {M | monthly} | {Y | yearly} | none}]
[--verbose=Boolean(false)]
```

Options

`--after=`*String*
: Only show backups created after this date. *String* must be in the format of YYYY-MM-DD.

`--before=`*String*
: Only show backups created before this date. *String* must be in the format of YYYY-MM-DD.

`--contained-vm-name=`*String*
: Specifies a virtual machine client within a VMware container or vApp. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--dir=`*String*
: Specifies a top-level parent folder of the backup from which to begin listing folders and files.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.
Use `mccli domain show` to return a list of valid domain names.

`--labelNum=`*Integer*
: Specifies the label number of the backup. This argument is required.
Use `backup show` without supplying `--labelNum` to return a list of backups with integer label numbers.

`--name=`*String*
: Specifies the client for which to show backups. This argument is required.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then `--domain` is ignored.
Use `mccli client show` to return a list of valid client names.

`--recursive=`*Boolean(false)*
: If `true`, then backup folders and files are recursively shown.

`--retention={{D | daily} | {W | weekly} | {M | monthly} | {Y | yearly} | none}`
: Specifies the extended retention types to show.
Both short form and long form`none` retention type values are allowed and can be mixed. For example, all of the following are valid:

- `--retention=D,weekly`
- `--retention=Daily,W,monthly`

The `--expiration`, `--extend-expiration`, and `--retention options` are mutually exclusive.

`--verbose=`*Boolean(false)*
If `true`, then detailed information is returned.

Event codes

22236	Client does not exist.
22504	Failed to retrieve the backups for a client.

Examples

This command returns a list of backups for MyClient:

`mccli backup show --name=clients/MyClient`

```
0,23000,CLI command completed successfully.
Created                LabelNum Size
---------------------- -------- ----------
2014-1-10 15:51:30 PST  2        4767841280
2014-1-10 15:00:23 PST  1        4750878720
```

This command returns backups for virtual machine TEST1, which resides inside VMware container 10.31.183.55/FO2:

`mccli backup show --name=/10.31.183.55/FO2 --verbose=true --contained-vm-name=TEST1 --xml`

```
<CLIOutput>
 <Results>
  <ReturnCode>0</ReturnCode>
  <EventCode>23000</EventCode>
  <EventSummary>CLI command completed successfully.</EventSummary>
 </Results>
 <Data>
  <Row>
   <Created>2014-07-14 06:08:21 UTC</Created>
   <LabelNum>1</LabelNum>
   <Size>4200757</Size>
   <Retention>N</Retention>
   <Label>MOD-134224603642992503ce8f11c007c26d73711fac</Label>
   <Plugin>Windows VMware Image</Plugin>
   <Expires>2014-09-12 06:14:35 UTC</Expires>
   <Files />
   <Server>Avamar</Server>
  </Row>
 </Data>
</CLIOutput>
```

This command returns backups for vApp 10.31.183.55/APP1:

`mccli backup show --name=/10.31.183.55/APP1 --verbose=true --xml`

```
<CLIOutput>
 <Results>
  <ReturnCode>0</ReturnCode>
  <EventCode>23000</EventCode>
  <EventSummary>CLI command completed successfully.</EventSummary>
 </Results>
 <Data>
  <Row>
   <Created>2014-07-14 06:42:46 UTC</Created>
   <LabelNum>1</LabelNum>
   <Size>4204699</Size>
   <Retention>N</Retention>
```

```
   <Label>MOD-134224603642992503ce8f11c007c26d73711fac</Label>
   <Plugin />
   <Expires>2014-09-12 06:49:40 UTC</Expires>
   <Files />
   <Server>Avamar</Server>
   <Networks>VM Network,Test2</Networks>
  </Row>
 </Data>
</CLIOutput>
```

backup validate

The `mccli backup validate` command initiates a validation of a backup.

Syntax

```
mccli backup validate --name=String
--labelNum=Integer --plugin=Integer
[--cmd=String [--cmd=String ...]
[--dest-client-domain=String] [--dest-client-name=String] [--
domain=String] [--location=String]
[--wait[=min]]
```

Options

`--cmd=`*String*

Specifies one ore more optional plug-in commands. You can supply multiple `--cmd` arguments, but each argument can only specify one plug-in command.
For example, this is valid `--cmd` syntax:
`--cmd="verbose=5" --cmd="throttle=5"`
However, this is not valid `--cmd` syntax:
`--cmd="verbose=5 throttle=5"`

`--dest-client-domain=`*String*

Used with `--dest-client-name` to validate a replicated backup (that is, a backup for a client in the `REPLICATE` domain) on an alternate client.
Specifies the Avamar server domain that contains the alternative client specified by the `--dest-client-name` argument.
Use `mccli domain show` to return a list of valid domain names.

`--dest-client-name=`*String*

When performing any redirected restore, specifies the destination client.
When performing a redirected restore of specific folders or files, you must also supply `--dest-dir`.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--dest-client-domain` option is ignored.
Use `mccli client show` to return a list of valid client names.

`--domain=`*String(/)*

Specifies the Avamar server domain that contains the client specified by `--name`.
Use `mccli domain show` to return a list of valid domain names.

`--labelNum=`*Integer*

Specifies the label number of the backup. This argument is required.
Use `backup show` without supplying `--labelNum` to return a list of backups with integer label numbers.

`--location=`*String*

Specifies the location that backups were replicated to.

`--name=`*String*

Specifies the client from which the backup was originally taken. This argument is required.

If you supply a fully qualified client name (for example, `/clients/MyClient`), then `--domain` is ignored.

Use `mccli client show` to return a list of valid client names.

`--plugin=`*Integer*

Specifies the plug-in ID for the restore. This argument is required.

Use `mccli plugin show` to return a list of valid plug-in IDs.

`--wait`[=*min*]

Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22236	Client does not exist.
22298	Request to validate is rejected.
22315	Client validate scheduled.
23009	Invalid plugin specified on the CLI.

checkpoint

The `mccli checkpoint` resource is used to manage Avamar server checkpoints.

checkpoint cancel-validate

The `mccli checkpoint cancel-validate` command cancels an active (currently running) checkpoint validation

Syntax

```
mccli checkpoint cancel-validate
```

Event codes

22615	A checkpoint validation was cancelled.
22616	No checkpoint validation is running.

checkpoint create

The `mccli checkpoint create` command creates a checkpoint. An MCS flush occurs as part of the checkpoint.

Syntax

```
mccli checkpoint create [--wait[=min]] [--
override_maintenance_scheduler=Boolean]
```

Options

`--wait`[=*min*]

Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown

sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

`-- override_maintenance_scheduler=`*Boolean*
If `true`, temporarily suspends the maintenance scheduler.

Event codes

22608	An Avamar server checkpoint was successfully created.
22609	An error occurred creating an Avamar server checkpoint.

checkpoint delete

The `mccli checkpoint delete` command permanently deletes a checkpoint from the Avamar server.

Syntax

```
mccli checkpoint delete --cptag=String
```

Options

`--cptag=`*String*
Specifies which checkpoint to delete. *String* must be a valid checkpoint ID. This argument is required.
Use `mccli checkpoint show` to return a list of valid checkpoint IDs.

Event codes

22610	An Avamar server checkpoint was successfully deleted.
22611	An Avamar server checkpoint was not successfully deleted.
22617	The specified checkpoint was not found.

checkpoint show

The `mccli checkpoint show` command lists all checkpoints with summary or detailed information.

Syntax

```
mccli checkpoint show [--verbose=Boolean(false)]
```

Options

`--verbose=`*Boolean(false)*
If `true`, then detailed checkpoint information is returned.
If `false`, then summary information is returned.

Event codes

22531	Unexpected exception occurred.

checkpoint validate

The `mccli checkpoint validate` command validates (performs an HFS check on) a checkpoint.

Syntax

```
mccli checkpoint validate --cptag=String
[--checktype={full | rolling}] [--
override_maintenance_scheduler=Boolean] [--wait[=min]]
```

Options

`--checktype={full | rolling}`
Constrains checkpoint validation to one or more of the following checkpoint types:

- `full`—Perform all HFS checks
- `rolling`—Perform rolling HFS check

`--cptag=`*String*
Specifies which checkpoint to validate. *String* must be a valid checkpoint ID. This argument is required.
Use `mccli checkpoint show` to return a list of valid checkpoint IDs.

`-- override_maintenance_scheduler=`*Boolean*
If `true`, temporarily suspends the maintenance scheduler.

`--wait`[=*min*]
Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22612	Starting to validate an Avamar server checkpoint.
22617	The specified checkpoint was not found.

client

The `mccli client` resource is used to manage backup client accounts on the Avamar server.

client add

The `mccli client add` command registers a new client with the MCS and adds it to the default group. The client need not be connected to the network. Registering a client allows you to subsequently define policies for that client, even if the client has not yet been activated.

Note

The `mccli client add` command cannot be used to add clients to the `REPLICATE` domain.

Syntax

```
mccli client add --name=String
[--changed-block-tracking=Boolean(false)] [--cid=String] [--
contact=String]
[--confirmed=Boolean(true)]  [--container-path=String] [--
datacenter=String]
[--dataset-domain=String(/)] [--dataset=String]
[--domain=String(/)] [--email=String]
[--encryption={High | Medium | None}] [--folder=String]
[--host=String] [--instance-display-name=String] [--instance-
uuid=String] [--location=String] [--max-active-jobs=Integer] [--
ostack-id=String] [--ostack-tenant-id=String]
[--override-encryption=Boolean(false)]
[--override-retention=Boolean(false)]
[--overtime-option={ALWAYS | NEVER | NEXT | NEXT_SUCCESS}]
[--overtime=Boolean(false)] [--pageable=Boolean(false)]
[--pageaddr=String] [--pageport=String] [--phone=String] [--recursive-
protection=Boolean(false)] [--retention-domain=String(/)] [--
retention=String]
[--type={normal | vcenter | proxy | vmachine | vcontainer | vapp}]
[--vcontainer-inclusion={dynamic | static}]
[--view-type={datastore | host-cluster | vm-template}]
[--virtual-center-name=String] [--virtual-center-port=Integer]
[--virtual-center-username=String --virtual-center-password=String]
```

Options

`--changed-block-tracking=`*Boolean(false)*

If `true`, when adding a virtual machine, VMware container, or vApp client, changed block tracking is enabled.

`--cid=`*String*

Use a CID during normal client creation. This is a reserved option and should be used only with help of EMC support.

`--confirmed=`*Boolean(true)*

If `true`, moves the already protected sub-containers to the container's current domain.

`--contact=`*String*

Specifies responsible party contact information.

`--container-path=`*String*

When adding a VMware container or vApp, specifies the inventory path to that container.

`--datacenter=`*String*

When adding a virtual machine client, VMware container or vApp, specifies a fully qualified datacenter name in vCenter. This argument is required.

`--dataset-domain=`*String(/)*

Specifies the Avamar server domain that contains the dataset specified by the `--dataset` argument.

Use `mccli domain show` to return a list of valid domain names.

`--dataset=`*String*

Specifies an alternative dataset this client will use for on-demand backups, or when the group dataset is overridden. *String* must be a valid dataset name.

Use `mccli dataset show` to return a list of valid dataset names.

`--domain=`*String(/)*

Specifies the domain for the new client.

`--email=`*String*

Specifies responsible party email address.

`--encryption={``High` | `Medium` | `None`}

Specifies the encryption method that the client should use when performing on-demand backups and restores, or when the group encryption method is overridden.

Note

The exact encryption technology and bit strength used for any given client/server connection depends on a number of factors, including the client platform and Avamar server version. The *EMC Avamar Product Security Guide* provides additional information.

`--folder=`*String*
When adding a virtual machine client, specifies the folder path for virtual machines in the datacenter.

`--host=`*String*
If adding a VMware container or vApp, and `--view-type=host-cluster`, this option specifies the host path inside the datacenter.

`--instance-display-name=`*String*
For OpenStack only (type=ostack-instance), OpenStack instance display name. It will be same as name , if not present.

`--instance-uuid=`*String*
For OpenStack only (type=ostack-instance), OpenStack Instance UUid.

`--location=`*String*
Specifies location information.

`--max-active-jobs=`*Integer*

`--name=`*String*
Specifies the new client name. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--ostack-id=`*String*
For OpenStack only (type=ostack-instance), OpenStack Id.

`--ostack-tenant-id=`*String*
For OpenStack only (type=ostack-instance), OpenStack Tenant Id.

`--override-encryption=`*Boolean(false)*
If `true`, then use the encryption method specified by `--encryption` instead of the group encryption method for scheduled backups.

`--override-retention=`*Boolean(false)*
If `true`, then use the retention policy specified by `--retention` instead of the group retention policy for scheduled backups.

`--overtime-option={ALWAYS | NEVER | NEXT | NEXT_SUCCESS}`
Specifies scheduled group backup overtime behavior:

- `ALWAYS`—scheduled group backups can always run past the schedule duration setting.
- `NEVER`—scheduled group backups can never run past the schedule duration setting.
- `NEXT`—only the next scheduled group backup can run past the schedule duration setting.
- `NEXT_SUCCESS`—scheduled group backups can run past the schedule duration setting until a successful backup is completed.

The default is `NEXT_SUCCESS`.

`--overtime=`*Boolean(false)*

If `true`, then client can exceed its backup window during scheduled backups.

`--pageable=`*Boolean(false)*

If `true`, then client can be paged for the purpose of initiating activation or picking up new backup or restore work.

`--pageaddr=`*String*

Specifies the IP address that the Avamar server can use to contact the client.

`--pageport=`*String*

Specifies the data port that the Avamar server can use to contact the client.

`--phone=`*String*

Specifies responsible party telephone number.

`--recursive-protection=`*Boolean(false)*

If true, recursively protects all clients (including child vm entities) for a contaner.

`--retention-domain=`*String(/)*

Specifies the Avamar server domain that contains the retention policy specified by the `--retention` argument.

Use `mccli domain show` to return a list of valid domain names.

`--retention=`*String*

Specifies an alternative retention policy this client will use for on-demand backups, or when the group retention policy is overridden.

`--type={normal | vcenter | proxy | vmachine | vcontainer | vapp}`

Specifies the type of client to add:

- `normal`—any client that is not one of the other types. This is the default client type.
- `vcenter`—vCenter client.
- `proxy`—Avamar proxy.
- `vmachine`—virtual machine client.
- `vcontainer`—VMware container.
- `vapp`—VMware vApp.

The default client type is `normal`.

`--vcontainer-inclusion={dynamic | static}`

When adding a VMware container or vApp, specifies the inclusion settings:

- `dynamic`—includes all contents of the vCenter container, but also continuously monitors the container entity in vCenter, so that if changes occur (for example, virtual machines or folders are added or deleted), those changes will be automatically reflected in Avamar.
- `static`—only includes what is in the vCenter container at the time it is added to Avamar. If subsequent changes occur in vCenter, they will not be reflected in Avamar.

`--view-type={datastore | host-cluster | vm-template}`

When adding a VMware container or vApp, specifies which view will be used to locate the container or vApp inside vCenter:

- `datastore`—Datastores view
- `host-cluster`—Hosts and Clusters view
- `vm-template`—Virtual Machines and Templates view

`--virtual-center-name=`*String*

When adding an Avamar proxy, specifies the vCenter name.

`--virtual-center-port=`*Integer*

When adding a vCenter client, specifies the vCenter port address.

`--virtual-center-username=`*String*

`--virtual-center-password=`*String*

When adding a vCenter client, specifies the vCenter administrative username and password.

Event codes

22210	Client successfully added.
22238	Client exists.
22263	Client registration error.
22288	Dataset does not exist.
22289	Retention policy does not exist.
22558	A domain or client with this name already exists.
23012	Invalid encryption method specified on the CLI.
30922	Failed to connect to vCenter.

Notes

The `--overtime` and `--overtime-option` arguments interact as follows:

- If you specify `--overtime=true` but do not specify an `--overtime-option`, then `--overtime-option` is automatically set to `NEXT_SUCCESS`.
- If you specify `--overtime=false` but do not specify an `--overtime-option`, then `--overtime-option` is automatically set to `NEVER`.
- If you supply both `--overtime` and `--overtime-option`, then `--overtime-option` takes precedence.

Examples

This command adds a new client called MyClient:

`mccli client add --name=MyClient`

```
0,22210,Client added
Attribute   Value
----------- ----------------------------------------
action      add
domain      /
node        MyClient
clientid    79a1042d7f5158c660fb7b863281f9787f8cb942
```

This command adds a new client called MyClient, and formats the output in XML:

`mccli client add --name=MyClient --xml`

```
<CLIOutput>
 <Results>
  <ReturnCode>0</ReturnCode>
  <EventCode>22210</EventCode>
  <EventSummary>Client successfully added</EventSummary>
 </Results>
 <Data>
  <Row>
   <Attribute>action</Attribute>
   <Value>add</Value>
  </Row>
  <Row>
   <Attribute>domain</Attribute>
   <Value>/</Value>
  </Row>
```

```
  <Row>
   <Attribute>node</Attribute>
   <Value>MyClient</Value>
  </Row>
  <Row>
   <Attribute>clientid</Attribute>
   <Value>70279698b5f755b1bc2f41432b3b3471048478e7</Value>
  </Row>
 </Data>
</CLIOutput>
```

This command adds a new proxy called MyProxy:

mccli client add --name=/clients/MyProxy --type=proxy --virtual-center-name=vcenter-1.example.com

```
0,22210,Client added
Attribute                      Value
------------------------------ ----------------------------------------
restoreOnly                    false
AllowClientOverrideSchedule    false
pagePort                       N/A
plugins                        <Plugins/>
registeredDate                 N/A
nodeName                       MyProxy
display-nodeName               MyProxy
DSOverride                     false
checkinDate                    N/A
windowsHardwareProfile         N/A
clientid                       f8888acc3c2fb8db604bd6a1abbacc5bd57b9a9d
encryptionMethod               high
pageAddress                    N/A
canPage                        true
backedUpDate                   N/A
action                         add
AllowFileSelectionOnSccBackups true
agentVersion                   unKnown
pageAddrLocked                 false
windowsID                      N/A
enabled                        false
isClientOs                     false
DATASETID                      Default:SNAPID
AllowClientAddToDataset        false
retryCnt                       2
overtimeOption                 NEXT_SUCCESS
RPOverride                     false
modifiedDate                   Thu Aug 25 04:45:21 UTC 2014
AllowSccBackups                true
nodeAddress                    N/A
timeOut                        10
registered                     false
POLICYID                       Default:POLICYID
overrideEncryption             false
OverrideStandardSccRetpol      false
fullName                       /clients/MyProxy
```

This command adds a new virtual machine client residing in the vcenter-1.example.com/VirtualMachines/Lab1 folder:

mccli client add --type=vmachine --name=new-vm --datacenter=Datacenter1 --domain=/vcenter-1.example.com/VirtualMachines --folder=Lab1

```
0,22210,Client added
Attribute                      Value
------------------------------ ----------------------------------------
restoreOnly                    false
```

```
AllowClientOverrideSchedule    false
pagePort                       N/A
plugins                        <Plugins><Plugin Build="ALL" Description="Linux VMware Image"
Version="7.1.100"/></Plugins>
registeredDate                 N/A
nodeName                       new-vm_UgqlbhZtNPdDKnHEpQLMOQ
display-nodeName               new-vm
DSOverride                     false
checkinDate                    N/A
windowsHardwareProfile         N/A
clientid                       1bdcf9e5f67a0e383202bbbe307473a59131fdcc
encryptionMethod               high
pageAddress                    N/A
canPage                        true
backedUpDate                   N/A
action                         add
AllowFileSelectionOnSccBackups true
agentVersion                   unKnown
pageAddrLocked                 false
windowsID                      N/A
enabled                        true
isClientOs                     false
DATASETID                      VMWARE:SNAPID
AllowClientAddToDataset        false
retryCnt                       2
overtimeOption                 NEXT_SUCCESS
RPOverride                     false
modifiedDate                   Thu Aug 25 05:06:38 UTC 2014
AllowSccBackups                true
nodeAddress                    N/A
timeOut                        10
registered                     false
POLICYID                       Default:POLICYID
overrideEncryption             false
OverrideStandardSccRetpol      false
fullName                       /vcenter-1.example.com/ACMCommunity_UgqlbhZtNPdDKnHEpQLMOQ
```

client add-datastore

The `mccli client add-datastore` command adds one or more new datastores to be protected by the specified Avamar proxy.

Syntax

```
mccli client add-datastore --datacenter=String
--datastore-name=String --name=String [--domain=String(/)] [--vcenter-
esx-name=String]
```

Options

`--datacenter=`*String*
: Specifies a fully qualified datacenter name in vCenter in the format of *path*|*name*. This argument is required.

`--datastore-name=`*String*
: Specifies the datastore name to be protected by the Avamar proxy. This argument is required.
Multiple `--datastore-name` arguments can be supplied with a single `mccli client add-datastore` command.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*
: Specifies the Avamar proxy to which the datastore will be added. String must be a valid Avamar proxy name. This argument is required.
Use `mccli client show` to return a list of valid client names.

If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--vcenter-esx-name=`*String*
Specifies the vcenter/esx fully qualified domain name to which the datastores need to be associated. Required if proxy has multiple vCenter-esx associations. If vCenter is under root domain, only vCenter name can be specified.

Event codes

22236	Client does not exist.
24003	Failed to update datastore mappings of a client.
24004	Datastore mappings of a client successfully updated.

Examples

This command adds datastore Storage1 and assigns proxy backupproxy225 to it:

```
mccli client add-datastore --name=/clients/backupproxy225
--datacenter=VAAYU-DEV-WIN --datastore-name=Storage1
```

```
0,24004,Datastore mappings of a client successfully updated.
Attribute   Value
----------- ----------------------
proxyclient /clients/backupproxy225
```

client backup dataset

The `mccli client backup dataset` command is used to initiate a client backups using a specified dataset.

Syntax

```
mccli client backup-dataset --name=String
[--cmd=String] [--contained-vm-name=String] [--dataset=String] [--
dataset-domain=String(/)]
[--domain=String(/)]
```

Options

`--cmd=`*String*
Specifies one ore more optional plug-in commands. You can supply multiple `--cmd` arguments, but each argument can only specify one plug-in command.
For example, this is valid `--cmd` syntax:
`--cmd="verbose=5" --cmd="throttle=5"`
However, this is not valid `--cmd` syntax:
`--cmd="verbose=5 throttle=5"`

`--contained-vm-name=`*String*
Specifies the VM/vAPP name inside the Container.

`--dataset=`*String*
Specifies the name of the dataset.

`--dataset-domain=`*String*
Specifies the domain of the dataset.

`--domain=`*String*
Specifies the domain of the client.

`--name=`*String*
Specifies the name of the client.

Event codes

22236	Client does not exist.
22305	Client backup scheduled.
23003	Invalid dataset name specified on the CLI.

Examples

This command schedules a client backup:

```
mccli client backup-dataset --name=/clients/vista32bit --dataset=13340
--cmd="verbose=5" --cmd="throttle=1"
```

```
0,22305,client backup scheduled.
```

client backup-group-dataset

The `mccli client backup-group-dataset` command initiates an on-demand backup of a single client using the group dataset. The client must already be a member of the group, or the backup fails.

This command initiates an on-demand client backup, not an on-demand group backup. Therefore, even if the client is a member of a group that is disabled and you specify the dataset for the group, the backup still occurs.

Syntax

```
mccli client backup-group-dataset --name= String
--group-name=String [--cmd=String [--cmd= ...]]
[--domain=String(/)] [--group-domain=String(/)] [--wait[=min]]
```

Options

`--cmd=`*String*

Specifies one ore more optional plug-in commands. You can supply multiple `--cmd` arguments, but each argument can only specify one plug-in command.
For example, this is valid `--cmd` syntax:
`--cmd="verbose=5" --cmd="throttle=5"`
However, this is not valid `--cmd` syntax:
`--cmd="verbose=5 throttle=5"`

`--domain=`*String(/)*

Specifies the Avamar server domain that contains the client specified by `--name`.

`--group-domain=`*String(/)*

String must be

`--group-name=`*String*

String must be

`--name=`*String*

Specifies which client to back up. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--wait`[=*min*]

Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22226 Group disabled.

22228 A client was not backed up because it is disabled, retired, or its plug-in(s) has backups disabled.

22234 Group does not exist.

22236 Client does not exist.

22241 Client is not a member of group.

22253 Client Adhoc Backup Request Error - Exception.

22305 Client backup scheduled.

client backup-target

The `mccli client backup-target` command initiates an on-demand backup of folders or files on a client.

This command returns an activity ID, which can be passed to `mccli activity show` to get status for this backup activity.

Note

The `mccli client backup-target` command cannot be used to back up folders or files belonging to clients in the `REPLICATE` domain.

Syntax

```
mccli client backup-target --name=String --target=String
[--cmd=String [--cmd= ...] [--contained-vm-name=String]
[--domain=String(/)] [--plugin=Integer] [--wait[=min]]
```

Options

`--cmd=`*String*

Specifies one ore more optional plug-in commands. You can supply multiple `--cmd` arguments, but each argument can only specify one plug-in command.
For example, this is valid `--cmd` syntax:
`--cmd="verbose=5" --cmd="throttle=5"`
However, this is not valid `--cmd` syntax:
`--cmd="verbose=5 throttle=5"`

`--contained-vm-name=`*String*

Specifies a virtual machine client within a VMware container or vApp. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--domain=`*String(/)*

Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*

Specifies which client to back up. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--plugin=`*Integer*

Specifies the plug-in ID.
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

`--target=`*String*

Specifies the folders or files to include in the backup.
At least one `--target` argument is required, and you can supply more than one `--target` on the same command line.
If backing up a virtual machine, individual disks can be backed up by specifying the corresponding base VMDK file. Multiple virtual disks can be backed up with additional `--target` arguments.
If backing up non-virtual filesystems, this argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--wait`[=*min*]
Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22228	A client was not backed up because it is disabled, retired, or its plug-in(s) has backups disabled.
22236	Client does not exist.
22253	Client Adhoc Backup Request Error - Exception.
22305	Client backup scheduled.
22309	Invalid plugin specified on the CLI.

Examples

This command backs up the Test1Windows222_1.vmdk virtual disk:

```
mccli client backup-target
--name=/vcenter-1.example.com/Test1Windows222 --plugin=3016
--target=[Storage2] Test1Windows222/Test1Windows222_1.vmdk
```

```
0,22305,client backup scheduled.
Attribute   Value
----------- --------------------------------------------------
client      /vcenter-1.example.com/Test1Windows222
target      [Storage2] Test1Windows222/Test1Windows222_1.vmdk
activity-id 9131407958056109
```

This command backs up a the TEST1 VMware container:

```
mccli client backup-target --name=/10.31.183.55/FO2
--contained-vm-name=TEST1 --plugin=3016 --target=ALL
```

```
0,22305,client backup scheduled.
Attribute    Value
------------ -------------------------------------------------
client       /10.31.183.55/FO2
target       ALL
contained-vm TEST1
activity-id  9134225778865209
```

client delete

The `mccli client delete` command permanently deletes a client and its backups from the Avamar server

Note

The `mccli client delete` command cannot be used to delete clients in the `REPLICATE` domain.

Syntax

```
mccli client delete --name=String [--domain=String(/)] [--delete-
child-vms=Boolean]
```

Options

`--delete-child-vms=`*Boolean*
Determines whether to delete the child vms that have other parents.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*
Specifies the client name. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

Event codes

22212	Client deleted.
22236	Client does not exist.
22240	Client delete failed.

client edit

The `mccli client edit` command edits the properties for a client.

There are no default settings for the mccli client edit command. If you enter this command but do not explicitly supply options and values on the command line, then there is no change to the client.

Note

The `mccli client edit` command cannot be used to edit clients in the `REPLICATE` domain.

Syntax

```
mccli client edit --name=String [--activated=Boolean(false)]
[--allow-cis-fileselection=Boolean(false)]
[--allow-cis=Boolean(false)] [--auto-datastore-mapping=Boolean]
[--changed-block-tracking=Boolean(false)] [--contact=String]
[--dataset-domain=String(/)] [--dataset=String]
[--domain=String(/)] [--email=String]
[--enabled=Boolean(false)]
[--encryption={High | Medium | None}]
```

```
[--location=String] [--max-active-jobs=integer] [--new-name=String]
[--override-cis-retention=Boolean(false)]
[--override-encryption=Boolean(false)]
[--override-retention=Boolean(false)]
[--overtime-option={ALWAYS | NEVER | NEXT | NEXT_SUCCESS}]
[--overtime=Boolean(false)] [--page-detection={Automatic | Manual}]
[--pageable=Boolean(false)] [--pageaddr=String]
[--pageport=String] [--phone=String]
[--recursive=Boolean] [--recursive-protection=Boolean] [--retention-
domain=String(/)] [--retention=String]
[--timeout=String] [--vcontainer-inclusion={dynamic | static}]
[--virtual-center-name=String] [--virtual-center-port=Integer]
[--virtual-center-username=String --virtual-center-password=String]
```

Options

`--activated=`*Boolean(false)*

If `true`, then client is assumed to have been previously activated.

`--allow-cis-fileselection=`*Boolean(false)*

If `true`, then users can make file selections when they initiate a backup of the client.

`--allow-cis=`*Boolean(false)*

If `true`, then users can initiate a backup of the client.

`--auto-datastore-mapping=`*Boolean*

If `true`, then changes the auto-datastore-mapping attribute for a registered proxy client. Option can be used only for a registered proxy during proxy client edit.

`--changed-block-tracking=`*Boolean(false)*

If `true`, when editing a virtual machine, VMware container, or vApp client, changed block tracking is enabled.

`--contact=`*String*

Specifies responsible party contact information.

`--dataset-domain=`*String(/)*

Specifies the Avamar server domain that contains the dataset specified by the `--dataset` argument.

Use `mccli domain show` to return a list of valid domain names.

`--dataset=`*String*

Specifies an alternative dataset this client will use for on-demand backups, or when the group dataset is overridden. *String* must be a valid dataset name.

Use `mccli dataset show` to return a list of valid dataset names.

`--domain=`*String(/)*

Specifies the domain for the new client.

`--email=`*String*

Specifies responsible party email address.

`--enabled=`*Boolean(false)*

If `true`, then client is eligible to immediately participate in on-demand and group backup and restore activities.

`--encryption={``High` | `Medium` | `None`}

Specifies the encryption method that the client should use when performing on-demand backups and restores, or when the group encryption method is overridden.

Note

The exact encryption technology and bit strength used for any given client/server connection depends on a number of factors, including the client platform and Avamar server version. The *EMC Avamar Product Security Guide* provides additional information.

`--location=`*String*
Specifies location information.

`--max-active-jobs=`*integer*
Specifies the maximum number of jobs this proxy can handle.

`--name=`*String*
Specifies which client to edit. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--new-name=`*String*
Specifies a new client name.

`--override-cis-retention=`*Boolean(false)*
If `true`, then this retention policy is used for all user-initiated backups of the client.

`--override-encryption=`*Boolean(false)*
If `true`, then use the encryption method specified by `--encryption` instead of the group encryption method for scheduled backups.

`--override-retention=`*Boolean(false)*
If `true`, then use the retention policy specified by `--retention` instead of the group retention policy for scheduled backups.

`--overtime-option=`{`ALWAYS` | `NEVER` | `NEXT` | `NEXT_SUCCESS`}
Specifies scheduled group backup overtime behavior:

- `ALWAYS`—scheduled group backups can always run past the schedule duration setting.
- `NEVER`—scheduled group backups can never run past the schedule duration setting.
- `NEXT`—only the next scheduled group backup can run past the schedule duration setting.
- `NEXT_SUCCESS`—scheduled group backups can run past the schedule duration setting until a successful backup is completed.

The default is `NEXT_SUCCESS`.

`--overtime=`*Boolean(false)*
If `true`, then client can exceed its backup window during scheduled backups.

`--page-detection=`{`Automatic` | `Manual`}
Specifies one of the following client page detection modes:

- `Automatic`—use automatic paging detection (that is, ignore `--pageaddr` and `--pageport` settings)
- `Manual`—use `--pageaddr` and `--pageport` settings

`--pageable=`*Boolean(false)*
If `true`, then client can be paged for the purpose of initiating activation or picking up new backup or restore work.

`--pageaddr=`*String*
Specifies the IP address that the Avamar server can use to contact the client.

`--pageport=`*String*

Specifies the data port that the Avamar server can use to contact the client.

`--phone=`*String*

Specifies responsible party telephone number.

`--recursive=`*Boolean*

If `true`, recursively updates all clients (including child vms) for a contaner.

`--recursive-protection=`*Boolean(false)*

If `true`, recursively protects all clients (including child vm entities) for a contaner.

`--retention-domain=`*String(/)*

Specifies the Avamar server domain that contains the retention policy specified by the `--retention` argument.

Use `mccli domain show` to return a list of valid domain names.

`--retention=`*String*

Specifies an alternative retention policy this client will use for on-demand backups, or when the group retention policy is overridden.

`--timeout=`*String*

Specifies the client browse timeout in seconds.

`--vcontainer-inclusion={dynamic | static}`

When adding a VMware container or vApp, specifies the inclusion settings:

- `dynamic`—includes all contents of the vCenter container, but also continuously monitors the container entity in vCenter, so that if changes occur (for example, virtual machines or folders are added or deleted), those changes will be automatically reflected in Avamar.
- `static`—only includes what is in the vCenter container at the time it is added to Avamar. If subsequent changes occur in vCenter, they will not be reflected in Avamar.

`--virtual-center-name=`*String*

When adding an Avamar proxy, specifies the vCenter name.

`--virtual-center-port=`*Integer*

When adding a vCenter client, specifies the vCenter port address.

`--virtual-center-username=`*String*

`--virtual-center-password=`*String*

When adding a vCenter client, specifies the vCenter administrative username and password.

Event codes

22211	Client was successfully updated.
22236	Client does not exist.
22239	Client update failed.
23003	Invalid dataset name specified on the CLI.
23005	Invalid retention policy name specified on the CLI.
23012	Invalid encryption method specified on the CLI.
30922	Failed to connect to vCenter.

Notes

The `--overtime` and `--overtime-option` arguments interact as follows:

- If you specify `--overtime=true` but do not specify an `--overtime-option`, then `--overtime-option` is automatically set to `NEXT_SUCCESS`.

- If you specify `--overtime=false` but do not specify an `--overtime-option`, then `--overtime-option` is automatically set to `NEVER`.
- If you supply both `--overtime` and `--overtime-option`, then `--overtime-option` takes precedence.

Examples

This command sets the data port for vCenter-1.example.com:

```
mccli client edit --name=/vCenter-1.example.com
--virtual-center-port=446
```

```
0,22211,Client was updated.
```

This command renames Proxy-1 to Proxy-2, and adds a contact phone number:

```
mccli client edit --name=/clients/Proxy1
--virtual-center-name=vCenter-1.example.com --new-name=Proxy2
--phone=555-1212
```

```
0,22211,Client was updated.
```

This command renames virtual machine client vm-1 to vm-2:

```
mccli client edit --name=/clients/vm-1 --new-name=clients/vm-2
```

```
0,22211,Client was updated.
```

This command sets dynamic inclusion for VMware container F02:

```
mccli client edit --name=/10.31.183.55/FO2
--vcontainer-inclusion=dynamic
```

```
0,22211,Client was updated.
```

This command turns on changed block tracking for VMware container F02:

```
mccli client edit --name=/10.31.183.55/FO2
--changed-block-tracking=true
```

```
0,22211,Client was updated.
```

client generate-sessiontoken

The `mccli client generate-sessiontoken` command is used to generate a session token for Avamar Desktop/Laptop.

Syntax

```
mccli client generate-sessiontoken --name=String [--domain=String(/)]
[--expiry=Integer]  [--fquser=String] [--islocaluser=Boolean(false)]
[--user=String]
```

Options

`--domain=`*String*
: Specifies the domain of the client.

`--expiry=`*Integer*
: Specifies the number of seconds after which token expires.

`--fquser=`*String*
: Specifies the fully qualified client user.

`--islocaluser=`*Boolean(false)*
: Specifies whether the user is local or not.

`--name=`*String*
Specifies the domain of the client.
`--user=`*String*
Specifies the client user.

client import-clients-from-file

The `mccli client import-clients-from-file` command reads a clients definition input file, then registers any clients defined in that clients definition input file.

Syntax

```
mccli client import-clients-from-file --file=String
[--verbose=Boolean(false)]
```

Options

`--file=`*String*
Specifies the full path and filename of the clients definition input file. The file must be in XML format. This argument is required.
The *EMC Avamar Administration Guide* provides additional information about clients definition files.
`--verbose=`*Boolean(false)*
If `true`, then a list of clients and their load status is output to stdout.

Event codes

22263	Client registration error.
22532	Client(s) successfully bulk-loaded.
23006	CLI client bulk load encountered one or more errors.

client invite

The `mccli client invite` command initiates activation of a specific client by first paging that client. If the client responds, then activation is completed and a Client ID (CID) is assigned to the client.

Note

The `mccli client invite` command cannot be used to invite clients to activate in the `REPLICATE` domain.

Syntax

```
mccli client invite --name=String [--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the client specified by `--name`.
`--name=`*String*
Specifies the client name. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

Event codes

22236	Client does not exist.
22237	Client invited to activate with server.
22263	Client registration error.
22271	Client registration error - unable to contact client on port.
22280	Client reconnect error - host name mismatch.
22282	Client reconnect error - unknown ID.
22295	Client registration error - server not available.

client move

The `mccli client move` command permanently moves a client and all its backups from one domain to another.

Note

The `mccli client move` command cannot be used to move clients in or out of the `REPLICATE` domain.

Syntax

```
mccli client move --name=String --new-domain=String
[--domain=String(/)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*
: Specifies which client to move. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--new-domain=`*String*
: Specifies the new domain for the client. This argument is required.

Event codes

22515	Client was moved.
22236	Client does not exist.
22501	Failed to move client.
23019	Client move failed - still active.

client remove-datastore

The `mccli client remove-datastore` command removes one or more datastores from the specified Avamar proxy.

Syntax

```
mccli client remove-datastore --datacenter=String
--datastore-name=String --name=String [--domain=String(/)] [--vcenter-
esx-name=String]
```

Options

`--datacenter=`*String*
: Specifies a fully qualified datacenter name in vCenter in the format of *path*|*name*. This argument is required.

`--datastore-name=`*String*
: Specifies the datastore name to be removed from the Avamar proxy. This argument is required.
Multiple `--datastore-name` arguments are allowed.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*
: Specifies the Avamar proxy from which the datastore will be removed. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--vcenter-esx-name=`*String*
: Specifies the vcenter/esx fully qualified domain name to which the datastores need to be associated. Required if proxy has multiple vCenter-esx associations. If vCenter is under root domain, only vCenter name can be specified.

Event codes

22236	Client does not exist.
24003	Failed to update datastore mappings of a client.
24004	Datastore mappings of a client successfully updated.

Examples

This command removes datastore Storage1 from client backupproxy225:

```
mccli client remove-datastore --name=/clients/backupproxy225
--datacenter=VAAYU-DEV-WIN --datastore-name=Storage1
```

```
0,24004,Datastore mappings of a client successfully updated.
Attribute   Value
----------- ----------------------
proxyclient /clients/backupproxy225
```

client retire

The `mccli client retire` command retires a client from active backup activities. All backups belonging to the client expire on the dates originally assigned to them.

Note

The `mccli client retire` command cannot be used to retire clients in the `REPLICATE` domain.

Syntax

```
mccli client retire --name=String
--expiration={YYYY-MM-DD | +nn{D | W | M | Y} | NO_EXPIRATION}
[--domain=String(/)] [--retire-child-vms=Boolean]
```

Options

`--expiration={`*YYYY-MM-DD*`|+`*nn*`{D|W|M|Y}|NO_EXPIRATION}`
Specifies an expiration date for backups belonging to the retired client. This argument is required.

- *YYYY-MM-DD*—Specifies an explicit expiration date.
- +*nn*{`D`|`W`|`M`|`Y`}—Specifies a duration from today. (For example, `+4W` specifies 4 weeks from today.)
- `NO_EXPIRATION`—Specifies that backups should never expire.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*
Specifies which client to retire. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--retire-child-vms=`*Boolean*
If `true`, retires the child WMs that have other parents.

Event codes

22236	Client does not exist.
22506	Failed to retire a client.
22512	Client was retired.

client show

The `mccli client show` command lists clients and their properties.

Syntax

```
mccli client show [--contained-vm-name=String]
[--contained-vms=Boolean(false)] [--domain=String(/)]
[--name=String] [--recursive=Boolean(false)]
[--replicated=Boolean(false)] [--retired=Boolean(false)]
[--verbose=Boolean(false)]
```

Options

`--contained-vm-name=`*String*
: Specifies a virtual machine client within a VMware container or vApp. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--contained-vms=`*Boolean(false)*
: If `true`, then shows details for VMware containers or vApps. This option is only valid if the client specified by `--name` is a VMware container or vApp.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.
If you supply `--retired`, then this argument is ignored.

`--name=`*String*
: Specifies a client name. If not supplied, all clients with the domain specified by `--domain=` are listed.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--recursive=`*Boolean(false)*
: If `true`, then the command shows all clients in the domain and all subdomains specified by `--domain`.
If `false` or not supplied, then the command only shows clients in the specified domain. Subdomains are not examined.

`--replicated=`*Boolean(false)*
: If `true`, then clients in the `REPLICATE` domain are shown.
If false or not supplied, then clients in the `REPLICATE` domain are not shown.

`--retired=`*Boolean(false)*
: If `true`, then retired clients are listed.
If supplied, then the `--domain` argument is ignored.

`--verbose=`*Boolean(false)*
: If you do not supply `--name` and this option is false or not supplied, then only the domain and client name are returned for each client.
If `true`, then other properties are also returned for each client.
This option is only meaningful when you do not supply the `--name` argument. When you supply`--name`, the details for the client are always returned.

Event codes

22236	Client does not exist.
22542	Domain does not exist.

Examples

This command returns a simple list of all clients in the clients domain:

```
mccli client show --domain=clients
```

```
0,23000,CLI command completed successfully.
command completed successfully
Name     Domain
-------- --------
MyClient /clients
```

This command returns a detailed list of properties for MyClient.example.com:

```
mccli client show --name=/clients/MyClient.example.com
```

```
0,23000,CLI command completed successfully.
command completed successfully
Attribute                                                 Value
--------------------------------------------------------- ----------------------------------
Name                                                      MyClient
Fully Qualified Name                                      /clients/MyClient.example.com
Operating System                                          Windows XP Professional SP 1.0
Paging                                                    Yes
Page Address                                              MyClient.example.com
Page Port                                                 28002
Paging Detection                                          Automatic
Agent Version                                             7.1.100-nnn
CID                                                       6e6241ae236c2ab58205f413f24d3d6e0
CID Assigned                                              2013-10-13 00:00:00 PDT
Activated                                                 Yes
Activated Date                                            2013-10-13 12:28:13 PDT
Disabled                                                  No
Agent Last Started                                        2014-4-107.2.100-nnn7.1.100.nnn
06:00:00 PST
Last Check-in                                             2014-4-10 11:00:59 PST
Override Group Retention                                  No
Retention Policy                                          Default Retention
Allow Overtime                                            No
Restore Only                                              No
Encryption Method                                         High
Override Encryption                                       No
Allow client initiated backups                            Yes
Allow file selection in client initiated backups          Yes
Override retention policy on client initiated backups     No
Contact Name                                              N/A
Contact Phone                                             N/A
Contact Email                                             N/A
Contact Location                                          N/A
Contact Notes                                             N/A
Member of Group                                           /Default Group
Plugin                                                    Windows File System(3001)
Initial Install Date                                      2014-4-10 12:28:14 PDT
Last Version Registered                                   2014-4-10 12:28:14 PDT
Last Successful Backup                                    2014-4-10 22:00:41 PST
```

This command shows detailed information for virtual machines in the F02 VMware container:

mccli client show --name=/10.31.183.55/FO2
--contained-vms=true --xml

```
<CLIOutput>
  <Results>
    <ReturnCode>0</ReturnCode>
    <EventCode>23000</EventCode>
    <EventSummary>CLI command completed successfully.</EventSummary>
  </Results>
  <Data>
    <Row>
      <Client>TEST1</Client>
      <Domain>/10.31.183.55/HleDynamicClients</Domain>
      <ClientType>Virtual Machine</ClientType>
    </Row>
    <Row>
      <Client>VA1</Client>
      <Domain>/10.31.183.55/HleDynamicClients</Domain>
      <ClientType>Virtual Container</ClientType>
    </Row>
  </Data>
</CLIOutput>
```

This command shows detailed information for virtual machine TEST1, which resides in VMware container F02:

mccli client show --name=/10.31.183.55/FO2
--contained-vm-name=TEST1

```
0,23000,CLI command completed successfully.
Attribute                Value
------------------------ ------------------------------------------------------------
Client Name              TEST1
Fully Qualified Name     /10.31.183.55/HleDynamicClients/TEST1_UCKZ3FGLz70WRRu3nq2ZuA
Client Type              Virtual Machine
Paging                   No
CID                      6214c7aac9ce678b0b19e4b05475d8fab58e2718
CID Assigned             2013-07-14 05:43:38 UTC
Disabled                 No
Agent Last Started       2013-07-14 05:43:38 UTC
Override Group Retention No
Retention Policy         Default Retention
Overtime Option          Overtime until successful backup
Restore Only             No
Encryption Method        High
Override Encryption      No
Allow CIS                Yes
Allow CIS file selection Yes
Override CIS retention   No
Browse Timeout           10 seconds
Contact Name             N/A
Contact Phone            N/A
Contact Email            N/A
Contact Location         N/A
Contact Notes            N/A
Member of Group          /10.31.183.55/Default Virtual Machine Group
Plugin                   Windows VMware Image(3016)
Initial Install Date     2013-07-14 05:43:38 UTC
Last Version Registered  2013-07-14 05:43:38 UTC
Last Successful Backup   N/A
```

client show-datastore

The `mccli client show-datastore` command returns detailed information for a datastore protected by a specific Avamar proxy.

Syntax

```
mccli client show-datastore --name=String [--domain=String(/)] --
vcenter-esx-name=String]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*
: Specifies the Avamar proxy that is protecting the datastore of interest. This argument is required.
Use `mccli client show` to return a list of valid proxy names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--vcenter-esx-name=`*String*
: Specifies the vcenter/esx fully qualified domain name to which the datastores need to be associated. Required if proxy has multiple vCenter-esx associations. If vCenter is under root domain, only vCenter name can be specified.

Event codes

22234 Group does not exist.

22236	Client does not exist.
24001	Failed to update proxy client mappings of a group.
24004	Proxy client mappings of a group successfully updated.

Examples

This command shows all datastores protected by client backupproxy225:

```
mccli client show-datastore --name=/clients/backupproxy225
```

```
Name     Type Accessible Hosts       Datacenter
-------- ---- ---------- ----------- -------------
Storage1 VMFS Yes        10.31.183.7 VAAYU-DEV-WIN
Storage2 VMFS Yes        10.31.183.7 VAAYU-DEV-WIN
```

client show-plugins

The `mccli client show-plugins` command lists all Avamar plug-ins installed on a client. If you also specify a plug-in number, then detailed information is returned for that plug-in.

Syntax

```
mccli client show-plugins --name=String
[--domain=String(/)] [--plugin=Integer] [--verbose=Boolean(false)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--name`.

`--name=`*String*
: Specifies the client name. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
: Specifies the plug-in ID.
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

`--verbose=`*Boolean(false)*
: If `true`, then detailed plug-in information is returned.

Event codes

22236	Client does not exist.
23009	Invalid plugin specified on the CLI.

client validate-clients-in-file

The `mccli client validate-clients-in-file` command validates a clients definition input file. The validation process ensures that the XML data is properly

formatted and that the file can be successfully processed by the `mccli client import-clients-from-file` command.

Syntax

```
mccli client validate-clients-in-file --file=String
[--verbose=Boolean(false)]
```

Options

`--file=`*String*
: Specifies the full path and filename of the clients definition input file. The file must be in XML format. This argument is required.
The *EMC Avamar Administration Guide* provides additional information about clients definition files.

`--verbose=`*Boolean(false)*
: If `true`, then a list of clients and their load status is output to stdout.

Event codes

23006	CLI client bulk load encountered one or more errors.

dataset

The `mccli dataset` resource is used to manage backup datasets on the Avamar server.

dataset add

The `mccli dataset add` command creates a dataset. By default, the initial dataset uses all available source data plug-ins and contains no explicit exclusion or inclusion entry entries. Use other `mccli dataset` commands to modify dataset properties after it has been created.

Syntax

```
mccli dataset add --name=String
[--alldata=Boolean(true)] [--domain=String(/)]
```

Options

`--alldata=`*Boolean(true)*
: If `true`, then the initial dataset uses all available source data plug-ins and contains no explicit exclusion or inclusion entries.
If `false`, then the initial dataset is empty.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
: Specifies the name of the new dataset. This argument is required.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

Event codes

22219	Dataset created.

23008	Dataset already exists.

dataset add-exclude

The `mccli dataset add-exclude` command adds an exclusion entry to a dataset.

Syntax

```
mccli dataset add-exclude --name=String
{--exclude=String | --exclude-file=String} --plugin=Integer
[--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--exclude=`*String*
Specifies folders or files to exclude from the dataset.
You must supply either `--exclude` or `--exclude-file`.
Multiple `--exclude=` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--exclude-file=`*filename*
Specifies the full path of a text *filename* that contains exclusion entries.
Each exclusion entry must conform to allowable `--exclude=` syntax and be on a single line.
You must supply either `--exclude` or `--exclude-file`.

`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new exclusion entry. This argument is required.
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.
23022	Error parsing input file.
23023	Input file could not be found.

dataset add-include

The `mccli dataset add-include` command adds an inclusion entry to a dataset.

Syntax

```
mccli dataset add-include--name=String
{--include=String | --include-file=String} --plugin=
[--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--include=`*String*
Specifies the folders or files to add back to the dataset after an exclusion entry has excluded them.
You must supply either `--include` or `--include-file`.
Multiple `--include` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--include-file=`*filename*
Specifies the full path of a text *filename* that contains inclusion entries.
Each inclusion entry must conform to allowable `--include` syntax and be on a single line.
You must supply either `--include` or `--include-file`.

`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new inclusion entry. This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.
23022	Error parsing input file.
23023	Input file could not be found.

dataset add-option

The `mccli dataset add-option` command adds a plug-in command (option) to a dataset.

Syntax

```
mccli dataset add-option --name=String
{--option=String [--option=String] ...}
{--value=String [--value=String] ...}
--plugin=Integer [--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--option=`*String*
`--value=`*String*
Specifies the plug-in command (option) name to add to the dataset. This argument is required.
Multiple `--option`/`--value` pairs are allowed.
Each client guide describes the valid plug-in commands for a specific dataset.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new plug-in command (option). This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

`--value=`*String*
Specifies a value for `--option=`. This argument is required.
Multiple `--option`/`--value` pairs are allowed.
Each client guide describes the valid plug-in commands for a specific dataset.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.

Notes
Plug-in commands and values, specified by the `--option` and `--value` arguments, must occur in equal numbers. Furthermore, the option-value pairing is positional. In other words, the first occurrence of `--option` is assigned the value specified by the first occurrence of `--value`, the second occurrence of `--option` is assigned the value specified by the second occurrence of `--value`, and so forth.

dataset add-target

The `mccli dataset add-target` command adds a target file or folder entry to a dataset.

Syntax

```
mccli dataset add-target --name=String --plugin=Integer
{--target=String | --target-file=String} [--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new target file or folder entry. This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

`--target=`*String*
Specifies a target file or folder to add to the dataset. This argument is required.
You must supply either `--target` or `--target-file`.

Multiple `--target` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--target-file=`*filename*
Specifies the full path of a text *filename* that contains target entries.
Each target entry must conform to allowable `--target` syntax and be on a single line.
You must supply either `--target` or `--target-file`.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.
23022	Error parsing input file.
23023	Input file could not be found.

dataset copy

The `mccli dataset copy` command copies a dataset, creating a new dataset.

Syntax

```
mccli dataset copy --name=String --new-name=String
[--domain=String(/)] [--new-domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
Specifies which dataset to copy. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--new-domain=`*String(/)*
Specifies the Avamar server domain where the new dataset should be created.

`--new-name=`*String*
Specifies the new dataset name. This argument is required.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--new-domain` argument is ignored.

Event codes

22219	Dataset created.
22288	Dataset does not exist.

dataset delete

The `mccli dataset delete` command permanently deletes a dataset from the Avamar server. You cannot delete a dataset if it is currently assigned to a client or group.

Syntax

```
mccli dataset delete --name=String [--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
Specifies which dataset to delete. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

Event codes

22221	Dataset deleted.
22288	Dataset does not exist.

dataset delete-exclude

The `mccli dataset delete-exclude` command permanently deletes exclusion entries from a dataset.

Syntax

```
mccli dataset delete-exclude --name=String
{--all | --exclude=String | --exclude-file=filename}
[--domain=String(/)] [--plugin=Integer]
```

Options

`--all`
Deletes all existing exclusion entries from the dataset.
You must supply `--all`, `--exclude` or `--exclude-file`.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--exclude=`*String*
Specifies the exclusion entry to delete from the dataset.
You must supply `--all`, `--exclude` or `--exclude-file`.
Multiple `--exclude=` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--exclude-file=`*filename*
Specifies the full path of a text *filename* that contains exclusion entries.
Each exclusion entry must conform to allowable `--exclude=` syntax and be on a single line.
You must supply `--all`, `--exclude` or `--exclude-file`.

`--name=`*String*
Specifies which dataset to modify. This argument is required.

Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new exclusion entry. This argument is required.
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.
23022	Error parsing input file.
23023	Input file could not be found.

dataset delete-include

The `mccli dataset delete-include` command permanently deletes inclusion entries from a dataset.

Syntax

```
mccli dataset delete-include --name=String --plugin=Integer
{--all | --include=String | --include-file=filename}
[--domain=String(/)]
```

Options

`--all`
Deletes all existing inclusion entries from the dataset.
`--all`, `--include` or `--include-file` are mutually exclusive. One of those arguments must be supplied with each command.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--include=`*String*
Specifies the folders or files to add back to the dataset after an exclusion entry has excluded them.
`--all`, `--include` or `--include-file` are mutually exclusive. One of those arguments must be supplied with each command.
Multiple `--include=` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--include-file=`*filename*
Specifies the full path of a text *filename* that contains inclusion entries.
Each inclusion entry must conform to allowable `--include` syntax and be on a single line.
`--all`, `--include` or `--include-file` are mutually exclusive. One of those arguments must be supplied with each command.

`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new inclusion entry. This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.
23022	Error parsing input file.
23023	Input file could not be found.

dataset delete-option

The `mccli dataset delete-option` command permanently deletes a plug-in command (option) from a dataset

Syntax

```
mccli dataset delete-option --name=String
{--option=String [--option=String] ...}
{--value=String [--value=String] ...}
--plugin=Integer [--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.
`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.
`--option=`*String*
`--value=`*String*
Specifies the plug-in command (option) name to delete from the dataset. This argument is required.
Multiple `--option`/`--value` pairs are allowed.
Each client guide describes the valid plug-in commands for a specific dataset.
`--plugin=`*Integer*
Specifies which plug-in ID gets the new plug-in command (option). This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.
`--value=`*String*
Specifies a value for `--option=`. This argument is required.
Multiple `--option`/`--value` pairs are allowed.
Each client guide describes the valid plug-in commands for a specific dataset.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.

23009	Invalid plugin specified on the CLI.

dataset delete-target

The `mccli dataset delete-target` command permanently deletes a target file or folder from an existing dataset.

Syntax

```
mccli dataset delete-target --name=String --plugin=Integer
{--target=String | --target-file=String} [--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new target file or folder entry. This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

`--target=`*String*
Specifies an existing target file or folder to delete from the dataset. This argument is required.
You must supply either `--target` or `--target-file`.
Multiple `--target` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--target-file=`*filename*
Specifies the full path of a text *filename* that contains target entries.
Each target entry must conform to allowable `--target` syntax and be on a single line.
You must supply either `--target` or `--target-file`.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.
23022	Error parsing input file.
23023	Input file could not be found.

dataset edit-option

The `mccli dataset edit-option` command modifies existing dataset options.

Syntax

```
mccli dataset edit-option --name=String
{--option=String [--option=String] ...}
{--value=String [--value=String] ...}
--plugin=Integer [--domain=String(/)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
: Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--option=`*String*

`--value=`*String*
: Specifies which plug-in command (option) name to modify. This argument is required.
Multiple `--option`/`--value` pairs are allowed.
Each client guide describes the valid plug-in commands for a specific dataset.

`--plugin=`*Integer*
: Specifies the plug-in ID. This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

`--value=`*String*
: Specifies a value for `--option=`. This argument is required.
Multiple `--option`/`--value` pairs are allowed.
Each client guide describes the valid plug-in commands for a specific dataset.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.

Notes

Plug-in commands and values, specified by the `--option` and `--value` arguments, must occur in equal numbers. Furthermore, the option-value pairing is positional. In other words, the first occurrence of `--option` is assigned the value specified by the first occurrence of `--value`, the second occurrence of `--option` is assigned the value specified by the second occurrence of `--value`, and so forth.

dataset replace

The `mccli dataset replace` command replaces all exclusion and inclusion entries for a dataset with the entries that you supply on the command line.

Syntax

```
mccli dataset add-exclude --name=String --plugin=Integer
[{--exclude=String | --exclude-file=String}]
[{--include=String | --include-file=String}]
[--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--exclude=`*String*
Specifies folders or files to exclude from the dataset.
You must supply either `--exclude` or `--exclude-file`.
Multiple `--exclude=` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--exclude-file=`*filename*
Specifies the full path of a text *filename* that contains exclusion entries.
Each exclusion entry must conform to allowable `--exclude=` syntax and be on a single line.
You must supply either `--exclude` or `--exclude-file`.

`--include=`*String*
Specifies the folders or files to add back to the dataset after an exclusion entry has excluded them.
You must supply either `--include` or `--include-file`.
Multiple `--include` arguments are allowed.
This argument accepts regular expression (regex) pattern matching operators, also known as wildcards.

`--include-file=`*filename*
Specifies the full path of a text *filename* that contains inclusion entries.
Each inclusion entry must conform to allowable `--include` syntax and be on a single line.
You must supply either `--include` or `--include-file`.

`--name=`*String*
Specifies which dataset to modify. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--plugin=`*Integer*
Specifies which plug-in ID gets the new entries. This argument is required
Use `mccli plugin show` to return a list of valid numeric plug-in IDs.

Event codes

22220	Dataset modified.
22288	Dataset does not exist.
23009	Invalid plugin specified on the CLI.

23022	Error parsing input file.
23023	Input file could not be found.

dataset show

The `mccli dataset show` command lists all datasets with summary information, or detailed information for a specific dataset.

Syntax

```
mccli dataset show [--domain=String(/)] [--name=String]
[--recursive=Boolean(false)] [--verbose=Boolean(false)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the dataset specified by `--name`.

`--name=`*String*
: Specifies a dataset name. If not supplied, all datasets are listed with summary information.
Use `mccli dataset show` to return a list of valid dataset names.
If you supply a fully qualified dataset name (for example, `/MyDomain/MyDataset`), then the `--domain` argument is ignored.

`--recursive=`*Boolean(false)*
: If `true`, then the command shows all datasets in the domain and any subdomains specified by the `--domain` argument.
If `false` or not supplied, then the command only shows datasets in the specified domain. Subdomains are not examined.

`--verbose=`*Boolean(false)*
: If `true`, then detailed dataset information is returned.
If `false` or not supplied, then only dataset names are returned.
If you also supply `--name`, then detailed information for that dataset is returned.

Event codes

22288	Dataset does not exist.
22542	Domain does not exist.

Examples

This command returns a simple list of all datasets currently in use:

`mccli dataset show`

```
0,23000,CLI command completed successfully.
command completed successfully
Dataset         Domain
--------------- ------
Unix Dataset    /
Windows Dataset /
Default Dataset /
Base Dataset    /
```

This command returns detailed information for all datasets currently in use:

`mccli dataset show --verbose=true`

```
0,23000,CLI command completed successfully.
command completed successfully
```

```
Dataset          Domain Num Targets Num Includes Num Excludes Num Options
---------------- ------ ----------- ------------ ------------ -----------
Unix Dataset     /      4           0            12           0
Windows Dataset  /      1           0            0            0
Default Dataset  /      6           0            0            0
Base Dataset     /      0           0            0            0
```

This command :

This command returns detailed information for a single dataset:

```
0,23000,CLI command completed successfully.
command completed successfully
Attribute                     Value
----------------------------- ------------
Name                          Unix Dataset
ID                            UNIX:SNAPID
Domain                        /
ReadOnly                      false
ALLDATA                       true
Num Targets                   4
Linux File System Target      ALL
Linux File System Target ID   1001
Solaris File System Target    ALL
Solaris File System Target ID 2001
HP-UX File System Target      ALL
HP-UX File System Target ID   4001
AIX File System Target        ALL
AIX File System Target ID     5001
Num Includes                  0
Num Excludes                  12
Linux File System Exclude     /tmp
Linux File System Exclude     /var/tmp
Linux File System Exclude     /usr/tmp
Linux File System Exclude     core
Linux File System Exclude     *cache.dat
Linux File System Exclude     *scan.dat
Solaris File System Exclude   /tmp
Solaris File System Exclude   /var/tmp
Solaris File System Exclude   /usr/tmp
Solaris File System Exclude   core
Solaris File System Exclude   *cache.dat
Solaris File System Exclude   *scan.dat
Num Options                   0
Is Link                       false
```

dd

The `mccli dd` resource is used to manage Data Domain systems for use as Avamar backup targets.

dd add

The `mccli dd add` command adds a new Data Domain system to an Avamar server.

Syntax

```
mccli dd add --name=String --max-streams=Integer
--user-name=String --password=String --password-confirm=String
[--default-storage=Boolean(false)] [--force]
[--rw-community=String] [--snmp-port=Integer(161)]
[--trap-port=Integer(163)] [--default-target=Boolean(false)] [--
instant-access-limit=Integer(1)]
```

Options

`--default-storage=`*Boolean(false)*

If `true` and the Data Domain system is a replication target, then use this Data Domain system as the default storage for data that replicates from the source Data Domain system.

`--default-target=`*Boolean(false)*

If `true`, indicates that the system identifed `--name=` should be used as the default target for Avamar backups.

`--force`

If another Data Domain system is currently designated as default storage on the Avamar server, and you want this Data Domain system to be the new default storage, you must supply `--force` with `--default-storage` to override the current default storage setting (that is, force the change to occur).

`--instant-access-limit=`*Integer(1)*

Specifies the maximum number of instant access sessions can be performed.

`--max-streams=`*Integer*

Specifies the maximum number of Data Domain system streams that Avamar uses for backups, restores, and replication. This argument is required.

`--name=`*String*

Specifies the fully qualified Data Domain system name or IP address. This argument is required.

`--rw-community=`*String*

Specifies the SNMP community for Avamar to have read/write access to the Data Domain system.

`--snmp-port=`*Integer(161)*

Specifies the data port number on the Data Domain system from which to receive and on which to set SNMP objects.

`--trap-port=`*Integer(163)*

Specifies the data port number on the Avamar server to which SNMP trap messages are sent.

`--user-name=`*String*

`--password=`*String*

`--password-confirm=`*String*

Specifies the username and password of the Data Domain OST account that Avamar will use to access the Data Domain system for backups, restores, and replication. These arguments are required.

Event codes

30934	Added Data Domain system.
30937	Failed to add Data Domain system.
30940	Data Domain system already exists.
30941	Failed to connect to Data Domain system.
30942	Unsupported DDOS version.
30948	The minimum number of allowed streams was exceeded.
30949	The maximum number of allowed streams was exceeded.
30953	Added Data Domain system but unable to retrieve the Data Domain system ssh key file.
30958	The DDBoost user is not set.

30959	The OST is disabled.
30960	DDBoost user is disabled.
30961	Failed to authenticate with ssh key file.
30962	Unable to add ssh public key for user.
30963	User does not exist.
30964	User is disabled.
30965	The user is not an admin user.
31004	Data Domain system added but the hostname may not be valid.

dd delete

The `mccli dd delete` command deletes a Data Domain system from an Avamar server.

Syntax

```
mccli dd delete --name=String [--force=Boolean(false)]
```

Options

`--name=`*String*
: Specifies which Data Domain system to delete. This argument is required.

`--force=`*Boolean(false)*
: If `true`, then forces deletion of the Data Domain system even if the system cannot be reached, has backups or checkpoints, or is the default replication storage system.

Event codes

30936	Deleted Data Domain system.
30939	Failed to delete Data Domain system.
30941	Failed to connect to Data Domain system.
30943	Data Domain system does not exist.

dd edit

The `mccli dd edit` command enables you to edit the Data Domain system configuration.

Syntax

```
mccli dd edit --name=String
[--default-storage=Boolean(false)] [--force] [--max-streams=Integer]
[--new-name=String] [--rw-community=String]
[--snmp-port=Integer(161)] [--trap-port=Integer(163)]
[--user-name=String --password=String --password-confirm=String]  [--
default-target=Boolean(false)] [--instant-access-limit=Interger(1)]
```

Options

`--default-storage=`*Boolean(false)*

If `true` and the Data Domain system is a replication target, then use this Data Domain system as the default storage for data that replicates from the source Data Domain system.

`--default-target=`*Boolean(false)*
If `true`, indicates that the system identifed `--name=` should be used as the default target for Avamar backups.

`--force`
If another Data Domain system is currently designated as default storage on the Avamar server, and you want this Data Domain system to be the new default storage, you must supply `--force` with `--default-storage` to override the current default storage setting (that is, force the change to occur).

`--instant-access-limit=`*Integer(1)*
Specifies the maximum number of instant access sessions can be performed.

`--max-streams=`*Integer*
Specifies the maximum number of Data Domain system streams that Avamar uses for backups, restores, and replication.

`--name=`*String*
Specifies the fully qualified Data Domain system name or IP address. This argument is required.

`--new-name=`*String*
Specifies a new fully qualified Data Domain system name.

`--rw-community=`*String*
Specifies the SNMP community for Avamar to have read/write access to the Data Domain system.

`--snmp-port=`*Integer(161)*
Specifies the data port number on the Data Domain system from which to receive and on which to set SNMP objects.

`--trap-port=`*Integer(163)*
Specifies the data port number on the Avamar server to which SNMP trap messages are sent.

`--user-name=`*String*

`--password=`*String*

`--password-confirm=`*String*
Specifies the username and password of the Data Domain OST account that Avamar will use to access the Data Domain system for backups, restores, and replication.

Event codes

30935	Updated Data Domain system in persistent store.
30938	Failed to update Data Domain system in persistent store.
30941	Failed to connect to Data Domain system.
30943	Data Domain system does not exist.
30948	The minimum number of allowed streams was exceeded.
30949	The maximum number of allowed streams was exceeded.
30953	Added Data Domain system but unable to retrieve the Data Domain system ssh key file.
30958	The DDBoost user is not set.
30959	The OST is disabled.
30960	DDBoost user is disabled.

30961 Failed to authenticate with ssh key file.

30962 Unable to add ssh public key for user.

30963 User does not exist.

30964 User is disabled.

30965 The user is not an admin user.

31005 Data Domain system added but the hostname may not be valid.

dd show-prop

The `mccli dd show-prop` command shows Data Domain system properties.

Syntax

```
mccli dd show-prop [--name=String]
```

Options

`--name=`*String*

Specifies the fully qualified Data Domain system name or IP address.
If supplied, system properties are returned for that Data Domain system.
If not supplied, a summary report of properties for all known Data Domain systems is returned.

Event codes

30943 Data Domain system does not exist.

Examples

This command returns system properties for a single Data Domain system (datadomain1.example.com):

`mccli dd show-prop --name=datadomain1.example.com`

```
0,23000,CLI command completed successfully.
Attribute                                  Value
------------------------------------------ ----------------------------------------
IPv4 Hostname                              datadomain1.example.com
IPv6 Hostname                              N/A
Total Capacity (post-comp size)            7623.8 GiB
Server Utilization (post-comp use%)        6%
Bytes Protected                            0 bytes
File System Available (post-comp avail)    7189.0 GiB
File System Used (post-comp used)          434.8 GiB
User Name                                  customer1
Default Replication Storage System         Yes
Target For Avamar Checkpoint Backups       No
Maximum Streams For Avamar Checkpoint Backups 0
Maximum Streams                            5
Maximum Streams Limit                      90
Instant Access Limit                       1
DDOS Version                               5.5.0.4-430231
Serial Number                              3FA0924231
Model Number                               DD670
Encryption Strength                        none
Authentication Mode                        none
Monitoring Status                          The Avamar server's time is ahead of DD's.
DDBoost Licensed                           true
DDBoost Enabled                            true
DDBoost User Enabled                       true
```

```
DDBoost User Status                              User Valid
DDBoost Option Status                            Option Enabled
SNMP status                                      Enabled
File System Status                               File System Running
Synchronization of maintenance operations        on
```

This command returns system properties for all Data Domain systems known to this MCS in XML format:

mccli dd show-prop --xml

```
Listening for transport dt_socket at address: 8003
<CLIOutput>
 <Results>
  <ReturnCode>0</ReturnCode>
  <EventCode>23000</EventCode>
  <EventSummary>CLI command completed successfully.</EventSummary>
 </Results>
 <Data>
  <Row>
   <IPv4Hostname>10.110.215.208</IPv4Hostname>
   <IPv6Hostname>N/A</IPv6Hostname>
   <TotalCapacity>7623.8 GiB</TotalCapacity>
   <ServerUtilization>6%</ServerUtilization>
   <BytesProtected>0 bytes</BytesProtected>
   <FileSystemAvailable>7189.0 GiB</FileSystemAvailable>
   <FileSystemUsed>434.8 GiB</FileSystemUsed>
   <UserName>ost</UserName>
   <DefaultReplicationStorageSystem>Yes</DefaultReplicationStorageSystem>
   <TargetForAvamarCheckpointBackups>No</TargetForAvamarCheckpointBackups>
   <MaximumStreamsForAvamarCheckpointBackups>0</MaximumStreamsForAvamarCheckpointBackups>
   <MaximumStreams>5</MaximumStreams>
   <MaximumStreamsLimit>90</MaximumStreamsLimit>
   <InstantAccessLimit>1</InstantAccessLimit>
   <DDOSVersion>5.5.0.4-430231</DDOSVersion>
   <SerialNumber>3FA0924231</SerialNumber>
   <ModelNumber>DD670</ModelNumber>
   <EncryptionStrength>none</EncryptionStrength>
   <AuthenticationMode>none</AuthenticationMode>
   <MonitoringStatus>The Avamar server&amp;apos;s time is ahead of DD&amp;apos;s.</MonitoringStatus>
   <DDBoostLicensed>true</DDBoostLicensed>
   <DDBoostEnabled>true</DDBoostEnabled>
   <DDBoostUserEnabled>true</DDBoostUserEnabled>
   <DDBoostUserStatus>User Valid</DDBoostUserStatus>
   <DDBoostOptionStatus>Option Enabled</DDBoostOptionStatus>
   <SNMPstatus>Enabled</SNMPstatus>
   <FileSystemStatus>File System Running</FileSystemStatus>
   <Synchronizationofmaintenanceoperations>on</Synchronizationofmaintenanceoperations>
  </Row>
  <Row>
   <IPv4Hostname>10.31.228.154</IPv4Hostname>
   <IPv6Hostname>N/A</IPv6Hostname>
   <TotalCapacity>404.8 GiB</TotalCapacity>
   <ServerUtilization>0%</ServerUtilization>
   <BytesProtected>0 bytes</BytesProtected>
   <FileSystemAvailable>403.0 GiB</FileSystemAvailable>
   <FileSystemUsed>1.7 GiB</FileSystemUsed>
   <UserName>ost</UserName>
   <DefaultReplicationStorageSystem>No</DefaultReplicationStorageSystem>
   <TargetForAvamarCheckpointBackups>No</TargetForAvamarCheckpointBackups>
   <MaximumStreamsForAvamarCheckpointBackups>0</MaximumStreamsForAvamarCheckpointBackups>
   <MaximumStreams>5</MaximumStreams>
   <MaximumStreamsLimit>16</MaximumStreamsLimit>
   <InstantAccessLimit>1</InstantAccessLimit>
   <DDOSVersion>5.5.0.4-430231</DDOSVersion>
   <SerialNumber>AUDVN985S7AG2T</SerialNumber>
   <ModelNumber>DDVE</ModelNumber>
   <EncryptionStrength>none</EncryptionStrength>
   <AuthenticationMode>none</AuthenticationMode>
   <MonitoringStatus>The Avamar server&amp;apos;s time is ahead of DD&amp;apos;s.</MonitoringStatus>
```

```
   <DDBoostLicensed>true</DDBoostLicensed>
   <DDBoostEnabled>true</DDBoostEnabled>
   <DDBoostUserEnabled>true</DDBoostUserEnabled>
   <DDBoostUserStatus>User Valid</DDBoostUserStatus>
   <DDBoostOptionStatus>Option Enabled</DDBoostOptionStatus>
   <SNMPstatus>Enabled</SNMPstatus>
   <FileSystemStatus>File System Running</FileSystemStatus>
   <Synchronizationofmaintenanceoperations>on</Synchronizationofmaintenanceoperations>
  </Row>
 </Data>
</CLIOutput>
```

dd show-util

The `mccli dd show-util` command shows Data Domain system utilization statistics.

Syntax

```
mccli dd show-util [--name=String]
```

Options

`--name=`*String*
: Specifies the fully qualified Data Domain system name or IP address.
If supplied, utilization statistics are returned for that Data Domain system.
If not supplied, utilization statistics for all known Data Domain systems is returned.

Event codes

30943	Data Domain system does not exist.

domain

The `mccli domain` resource is used to manage Avamar server server domains and subdomains.

domain add

The `mccli domain add` command creates a new domain or subdomain on the Avamar server.

Syntax

```
mccli domain add --name=String
[--contact=String] [--domain=String(/)]
[--email=String] [--location=String]
[--phone=String]
```

Options

`--contact=`*String*
: Specifies responsible party contact information.

`--domain=`*String(/)*
: Specifies the parent domain for the new domain.

`--email=`*String*
: Specifies responsible party email address.

`--location=`*String*
: Specifies location information.

`--name=`*String*

Specifies the name of the new domain. This argument is required.

`--phone=`*String*

Specifies responsible party telephone number.

Event codes

22526	Failed folder add.
22527	Domain added.
22540	Invalid name character.
22541	Domain already exists.
22558	A domain or client with this name already exists.

domain delete

The `mccli domain delete` command permanently deletes a domain from the Avamar server. Deleting a domain deletes the domain and all subdomains, clients, and backups stored in that domain.

Syntax

```
mccli domain delete --name=String [--domain=String(/)] [--
force=Boolean]
```

Options

`--domain=`*String(/)*

Specifies the parent domain for the new domain.

`--force=`*Boolean*

If `true`, forces a delete.

`--name=`*String*

Specifies which domain to delete. This argument is required.

Use `mccli domain show` to return a list of valid domain names.

If you supply a fully qualified domain name (for example, `/domain/subdomain`), then the `--domain` argument is ignored.

Event codes

22507	Failed domain deletion.
22513	Domain removed.
22542	Domain does not exist.
22543	Domain has groups.
22544	Domain has datasets.
22545	Domain has event profiles.
22627	Domain has schedules.
22628	Domain has retentions.

domain edit

The `mccli domain edit` command edits the properties for a domain.

Syntax

```
mccli domain edit --name=String
[--contact=String] [--domain=String(/)]
[--email=String] [--location=String]
[--phone=String]
```

Options

`--contact=`*String*
: Specifies responsible party contact information.

`--domain=`*String(/)*
: Specifies the parent domain.

`--email=`*String*
: Specifies responsible party email address.

`--location=`*String*
: Specifies location information.

`--name=`*String*
: Specifies which domain to edit. This argument is required.

`--phone=`*String*
: Specifies responsible party telephone number.

Event codes

22519	Domain modified.
22542	Domain does not exist.

domain show

The `mccli domain show` command displays properties for a domain.

Syntax

```
mccli domain show --name=String [--domain=String(/)]
[--recursive=Boolean(false)]
```

Options

`--domain=`*String(/)*
: Specifies the parent domain under which domains and subdomains are listed.

`--name=`*String*
: Specifies the name of the domain for which to show information. This argument is required.
  Use `mccli domain show` to return a list of valid domain names.
  If you supply a fully qualified domain name (for example, `/domain/subdomain`), then the `--domain` argument is ignored.

`--recursive=`*Boolean(false)*
: If `true`, then the command examines all domains in the domain and any subdomains.
  If `false` or not supplied, then the command only examines the domain. Subdomains are not examined.

Event codes

22542	Domain does not exist.

dump

The `mccli dump` resource dumps various DPNProxyService caches for troubleshooting purposes.

Note

The `mccli dump` resource is strictly reserved for EMC internal use only.

dump clientcache

The `mccli dump clientcache` command is used to dump the DPNProxyService client cache for troubleshooting purposes.

Note

The `mccli dump clientcache` command is strictly reserved for EMC internal use only.

Syntax

```
mccli dump clientcache
```

dump domaincache

The `mccli dump domaincache` command is used to dump the DPNProxyService domain cache for troubleshooting purposes.

Note

The `mccli dump domaincache` command is strictly reserved for EMC internal use only.

Syntax

```
mccli dump domaincache
```

dump jobcache

The `mccli dump jobcache` command is used to dump the DPNProxyService client job cache for troubleshooting purposes.

Note

The `mccli dump jobcache` command is strictly reserved for EMC internal use only.

Syntax

```
mccli dump jobcache
```

event

The `mccli event` resource is used to access and manage event codes on the Avamar server.

event ack

The `mccli event ack` command acknowledges events.

Syntax

```
mccli event ack {--all=Boolean(false)
| [--after=String] | [--before=String]
| [--category={APPLICATION | SECURITY | SYSTEM | USER}]
| [--domain=String(/)] | {[--exclude=String] | [--include=String]}
| [--id=Integer] | [--range=String]
| [--severity={NODE | NODE_FATAL | OK | PROCESS
| PROCESS_FATAL | SYSTEM_FATAL |  USER | USER_FATAL}]
| [--source={avamar | dd}]
| [--type={AUDIT | DEBUG | ERROR | INFORMATION | INTERNAL | WARNING}]}
```

Options

`--after=`*String*
: Acknowledges events after the specified date, which must be in the format of *YYYY-MM-DD HH:mm:ss*. The date value can be truncated as needed.

`--all=`*Boolean(false)*
: If `true`, then all events are acknowledged.

`--before=`*String*
: Acknowledges events before the specified date, which must be in the format of *YYYY-MM-DD HH:mm:ss*. The date value can be truncated as needed.

`--category={APPLICATION | SECURITY | SYSTEM | USER}`
: Acknowledges events of this category.

`--domain=`*String(/)*
: Acknowledges events of this Avamar server domain.

`--exclude=`*String*
: Specifies a comma-separated list of event codes to exclude from acknowledgment. The comma-separated list cannot contain spaces.
  The `--exclude` and `--include` options are mutually exclusive.

`--id=`*Integer*
: Acknowledges this single event.

`--include=`*String*
: Specifies a comma-separated list of event codes to include for acknowledgment. The comma-separated list cannot contain spaces.
  The `--exclude` and `--include` options are mutually exclusive.

`--range=`*String*
: Acknowledges events of this range of events. *String* must be in the format of *ID-1*:*ID-2*, where *ID-1* and *ID-2* are the lower and upper limits of the range, respectively.

`--severity={NODE | NODE_FATAL | OK | PROCESS | PROCESS_FATAL | SYSTEM_FATAL | USER | USER_FATAL}`
: Acknowledges events of this severity.

`--source={avamar | dd}`
: If `avamar` is specified, then the command acknowledges events for the Avamar server.

If`dd` is specified, then the command acknowledges events for all configured Data Domain systems.

`--type={AUDIT | DEBUG | ERROR | INFORMATION | INTERNAL | WARNING}`
Acknowledges events of this type.

Event codes

23022	Event id does not exist.

Notes

This command requires either the `--all` option, or some other filtering criteria.

The `--exclude` and `--include` options are mutually exclusive.

Output is a count of the events acknowledged.

Examples

This command acknowledges event code 148710:

```
mccli event ack --id=148710
```

```
0,23000,CLI command completed successfully.
Attribute    Value
------------ -----
events-acked 1
```

event clear-data-integrity-alerts

The `mccli event clear-data-integrity-alerts` command clears all data integrity alerts for this server.

This command requires a reset code, which must be obtained from EMC Technical Support.

Syntax

```
mccli event clear-data-integrity-alerts --reset-code=String
```

Options

`--reset-code=`*String*
Reset code provided by EMC Technical Support. This argument is required.

Event codes

23100	A data integrity alerts reset code was accepted.
23101	A data integrity alerts reset code was rejected.

event get-info

The `mccli event get-info` command returns detailed information for a specific event code.

Syntax

```
mccli event get-info --code=Integer
[--description=Boolean(false)] [--remedy=Boolean(false)]
[--summary=Boolean(true)]
```

Options

`--code=`*Integer*
Specifies the event code number. This argument is required.

`--description=`*Boolean(false)*
If `true`, then other properties are also returned, including the remedy and summary properties.
If `false` or not supplied, then only the numeric code is returned.

`--remedy=`*Boolean(false)*
If `true`, then event remedy information is returned.
If `false` or not supplied, then event remedy information is not returned.

`--summary=`*Boolean(true)*
If `true` or not supplied, then event summary information is returned.
If `false`, then event summary information is not returned.

event publish

The `mccli event publish` command is used to publish an event.

Syntax

```
mccli event publish --code=Integer [--attribute=String] [--
message=String] [--product=String(/)] [--value=String(/)]
```

Options

`--attribute=`*String*
Specifies the event data attribute.

`--integer=`*String*
Specifies the event code to publish.

`--message=`*String*
Specifies the event summary message.

`--product=`*String*
Specifies the product calling mccli: MCGUI, MCCLI, END_USER, WEB_RESTORE, EM, EMS, NONE, TEST, MCS, SNMP_SUB_AGENT, SCC, DTLT.

`--value=`*String*
Specifies the event data value.

Examples

This command publishes an event:

```
mccli event publish --code=99998 --message=Sandcrab Testing
13594-99998 --attribute=abcdefghi --value=0123456789 --product=MCCLI
```

```
23000,CLI command completed successfully.
Attribute Value
--------- ----------
abcdefghi 0123456789

mccli event show --include=99998 --after=2016-03-08 08:10:30

0,23000,CLI command completed successfully.
ID   Date                     Type  Code  Category Severity Domain
Summary
---- ------------------------ ----- ----- -------- -------- ------
---------------------------
9354 2016-03-08 08:10:32 IST ERROR 99998 SYSTEM   PROCESS  /
Sandcrab Testing 13594-99998
 Testing 13594-99998
```

event show

The `mccli event show` command returns event occurrence details.

Syntax

```
mccli event show
[--after=String] [--before=String]
[--category={APPLICATION | SECURITY | SYSTEM | USER}]
[--domain=String(/)] [--exclude=String]
[--id=Integer] [--include=String]
[--severity={NODE | NODE_FATAL | OK | PROCESS
| PROCESS_FATAL | SYSTEM_FATAL |  USER | USER_FATAL}]
[--source={avamar | dd}]
[--type={AUDIT | DEBUG | ERROR | INFORMATION | INTERNAL | WARNING}]}]
[--unack=Boolean(false)]
```

Options

`--after=`*String*
: Shows events after the specified date, which must be in the format of *YYYY-MM-DD HH:mm:ss*. The date value can be truncated as needed.

`--before=`*String*
: Shows events before the specified date, which must be in the format of *YYYY-MM-DD HH:mm:ss*. The date value can be truncated as needed.

`--category={APPLICATION | SECURITY | SYSTEM | USER}`
: Shows events of this category.

`--domain=`*String(/)*
: Shows events in this Avamar server domain.

`--exclude=`*String*
: Specifies a comma-separated list of event codes to exclude from the command output. The comma-separated list cannot contain spaces.
  The `--exclude` and `--include` options are mutually exclusive.

`--id=`*Integer*
: Shows information for this single event ID.

`--include=`*String*
: Specifies a comma-separated list of event codes to include in the command output. The comma-separated list cannot contain spaces.
  The `--exclude` and `--include` options are mutually exclusive.

`--severity={NODE | NODE_FATAL | OK | PROCESS | PROCESS_FATAL | SYSTEM_FATAL | USER | USER_FATAL}`
: Shows events with this severity.

`--source={avamar | dd}`
: If `avamar` is specified, then the command shows events for the Avamar server.
  If`dd` is specified, then the command shows events for all configured Data Domain systems.

`--type={AUDIT | DEBUG | ERROR | INFORMATION | INTERNAL | WARNING}`
: Shows events of this type.
  You can supply more than one `--type` option on a single command line.

`--unack=`*Boolean(false)*
: If `true`, then only unacknowledged events are returned.

Event codes

23022	Event id does not exist.

Notes

If multiple events are displayed, sorting is by date in descending order.

The `--exclude` and `--include` options are mutually exclusive.

Examples

This command returns detailed information for event code 149897:

```
mccli event show --id=149897
```

```
0,23000,CLI command completed successfully.
command completed successfully
Attribute   Value
----------- ------------------------------------------------------
ID          149897
Date        2014-04-19 14:19:45 PDT
Type        INFORMATION
Code        578
Category    SYSTEM
Severity    OK
Domain      /
Summary     hfscheckresults::merge, in checkpoint cp.20130419140741,
            this ((start 1176995040, nodestart 1176995040, nodefinish
            0, valid false, totalerrors 0)) tried to merge with
            ((start 0, nodestart 0, nodefinish 0, valid false,
            totalerrors 0)) but the uniquestarttime fields differ
SW Source   DPN:Unknown
For Whom    N/A
HW Source   node-10-0-54-249.example.com
Description N/A
Remedy      N/A
Notes       N/A
Data        <data><entry key="code" type="text" value="0578"
            version=""/><entry key="type" type="text" value="INFO"
            version=""/><entry key="time" type="text"
            value="21:19:45.93935" version=""/><entry key="message"
            type="text" value="hfscheckresults::merge, in checkpoint
            cp.20130419140741, this ((start 1176995040, nodestart
            1176995040, nodefinish 0, valid false, totalerrors 0))
            tried to merge with ((start 0, nodestart 0, nodefinish 0,
            valid false, totalerrors 0)) but the uniquestarttime
            fields differ" version=""/><entry key="date" type="text"
            value="2014/04/19" version=""/><entry key="thread"
            type="text" value="srvm-14070642#srv:265"
            version=""/></data>
```

group

The `mccli group` resource is used to manage groups and group policy.

group add

The `mccli group add` command creates a new group.

Syntax

```
mccli group add --name=String
--dataset=String(Default Dataset) --enabled=Boolean(false)
--retention=(Default Retention) --schedule=(Default Schedule)
[--dataset-domain=String(/)] [--domain=String(/)]
[--encryption={High | Medium | None}]
[--retention-domain=String(/)] [--schedule-domain=String(/)] [--auto-
proxy-mapping=Boolean(false)]
```

Options

`--auto-proxy-mapping=`*Boolean(false)*
if `true`, specifies that auto proxy mapping should be used.

`--dataset-domain=`*String(/)*
Specifies the Avamar server domain that contains the dataset specified by the `--dataset` argument.

`--dataset=`*String(Default Dataset)*
Specifies the dataset to assign to the group. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--enabled=`*Boolean(false)*
If `true`, then the group is eligible to immediately participate in scheduled backup activities. This argument is required .

`--encryption={High | Medium | None}`
Specifies the group default encryption method.

Note

The exact encryption technology and bit strength used for any given client/server connection depends on a number of factors, including the client platform and Avamar server version. The *EMC Avamar Product Security Guide* provides additional information.

`--name=`*String*
Specifies the new group name. This argument is required.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--retention-domain=`*String(/)*
Specifies the Avamar server domain that contains the retention policy specified by the `--retention` argument.

`--retention=`*(Default Retention)*
Specifies the retention policy to assign to the group. This argument is required.
Use `mccli retention show` to return a list of valid retention policy names.

`--schedule-domain=`*String(/)*
Specifies the Avamar server domain that contains the schedule specified by the `--schedule` argument.

`--schedule=`*(Default Schedule)*
Specifies the schedule to assign to the group. This argument is required.
Use `mccli schedule show` to return a list of valid schedule names.

Event codes

22207	New group created.
22233	Group already exists.
22235	Group add failed.
23003	Invalid dataset name specified on the CLI.
23004	Invalid schedule name specified on the CLI.
23005	Invalid retention policy name specified on the CLI.
23012	Invalid encryption method specified on the CLI.

31002 Group add or update failed due to a policy hierarchical management violation.

group add-client

The `mccli group add-client` command adds a client to a group.

Syntax

```
mccli group add-client --client-name=String --name=String
[--client-domain=String(/)] [--dataset-domain=String(/)]
[--dataset=String] [--domain=String(/)]
```

Options

`--client-domain=`*String(/)*
: Specifies the Avamar server domain that contains the client specified by `--client-name`.

`--client-name=`*String*
: Specifies which client to add to the group. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified dataset name (for example, `/clients/MyClient`), then the `--client-domain` argument is ignored.

`--dataset-domain=`*String(/)*
: Specifies the Avamar server domain that contains the dataset specified by the `--dataset` argument.

`--dataset=`*String*
: Specifies an alternative dataset this client will use when this group default dataset is overridden.
Use `mccli dataset show` to return a list of valid dataset names.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
: Specifies which group to modify. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

Event codes

22234	Group does not exist.
22236	Client does not exist.
22242	Client is already a member of this group.
22243	Client group membership successfully updated.
22269	Unable to add a client to a group.
22358	Unable to add a client to group due to incompatibility.

group add-proxy

The `mccli group add-proxy` command adds the specified Avamar proxy to the specified group.

Syntax

```
mccli group add-proxy --name=String --proxy-name=String
[--domain=String(/)] [--proxy-domain=String(/)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
: Specifies which group to modify. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--proxy-domain=`*String(/)*
: Specifies the Avamar server domain that contains the proxy specified by `--proxy-name`.

`--proxy-name=`*String*
: Specifies which proxy to add to the group. This argument is required.
Use `mccli client show` to return a list of valid proxy names.
If you supply a fully qualified proxy name (for example, `/clients/MyProxy`), then the `--proxy-domain` argument is ignored.

Event codes

22234	Group does not exist.
22236	Client does not exist.
24001	Failed to update proxy client mappings of a group.
24002	Proxy client mappings of a group successfully updated.

Examples

This command adds proxy backupproxy225 to the Test1 group:

```
mccli group add-proxy --domain=/vcenter-1.example.com --name=Test1
--proxy-domain=/clients --proxy-name=backupproxy225
```

```
0,24002,Proxy client mappings of a group successfully updated.
Attribute Value
--------- ---------------------------
group     /vcenter-1.example.com/Test1
```

group backup

The `mccli group backup` command initiates an on-demand group backup.

Syntax

```
mccli group backup --name=String [--domain=String(/)] [--wait[=min]]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
Specifies which group to back up. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--wait`[`=`*min*]
Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22226	Group disabled.
22227	Group does not contain any clients.
22228	A client was not backed up because it is disabled, retired or one or more of its plug-ins has backups disabled.
22234	Group does not exist.
22301	Scheduled group backups initiated for all clients.
22311	Scheduled group backups failed to start.
23028	Invalid group type.

group copy

The `mccli group copy` command copies an existing group, creating a new group. The new group inherits all the properties and settings of the original group except for client members. No clients are assigned to the new group.

Note

You must copy groups within the same domain. You cannot copy a group to another domain under any circumstances.

Syntax

```
mccli group copy --name=String --new-name=String
[--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
Specifies which group to copy. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--new-name=`*String*
Specifies the name for the new group. This argument is required.

Event codes

22207	New group created.
22233	Group already exists.
22234	Group does not exist.
22235	Group add failed.
22540	Invalid name character.

group delete

The `mccli group delete` command permanently deletes a group from the Avamar server.

A client must always be a member of at least one group. Therefore, if the group that you are deleting contains any clients that are not also members of at least one other group, then you must move those clients to other groups using the `mccli group move-client` command before you delete the group.

Syntax

```
mccli group delete --name=String [--domain=String(/)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
: Specifies which group to delete. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

Event codes

22209	Group successfully deleted.
22234	Group does not exist.
22247	Cannot delete the group because the client members are not present in other groups.

group edit

The `mccli group edit` command edits properties for a group.

There are no default settings for the mccli group edit command. If you enter the command but do not supply options and values on the command line, then no changes are made to the group.

Syntax

```
mccli group edit [--dataset-domain=String(/)]
[--dataset=String(Default Dataset)] [--domain=String(/)]
[--enabled=Boolean(false)]
[--encryption={High | Medium | None}]
```

```
[--name=String] [--retention-domain=String(/)]
[--retention=(Default Retention)] [--schedule-domain=String(/)]
[--schedule=(Default Schedule)] [--auto-proxy-mapping=Boolean(false)]
```

Options

`--auto-proxy-mapping=`*Boolean(false)*
: if `true`, specifies that auto proxy mapping should be used.

`--dataset-domain=`*String(/)*
: Specifies the Avamar server domain that contains the dataset specified by the `--dataset` argument.

`--dataset=`*String(Default Dataset)*
: Specifies the dataset to assign to the group. This argument is required.
Use `mccli dataset show` to return a list of valid dataset names.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the group specified by `--name`.

`--enabled=`*Boolean(false)*
: If `true`, then the group is eligible to immediately participate in scheduled backup activities. This argument is required .

`--encryption={High | Medium | None}`
: Specifies the group default encryption method.

> **Note**
>
> The exact encryption technology and bit strength used for any given client/server connection depends on a number of factors, including the client platform and Avamar server version. The *EMC Avamar Product Security Guide* provides additional information.

`--name=`*String*
: Specifies the group name. This argument is required.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--retention-domain=`*String(/)*
: Specifies the Avamar server domain that contains the retention policy specified by the `--retention` argument.

`--retention=`*(Default Retention)*
: Specifies the retention policy to assign to the group. This argument is required.
Use `mccli retention show` to return a list of valid retention policy names.

`--schedule-domain=`*String(/)*
: Specifies the Avamar server domain that contains the schedule specified by the `--schedule` argument.

`--schedule=`*(Default Schedule)*
: Specifies the schedule to assign to the group. This argument is required.
Use `mccli schedule show` to return a list of valid schedule names.

Event codes

22208	Group successfully modified.
22234	Group does not exist.
23003	Invalid dataset name specified on the CLI.
23004	Invalid schedule name specified on the CLI.

23005	Invalid retention policy name specified on the CLI.
23012	Invalid encryption method specified on the CLI.
31002	Group add or update failed due to a policy hierarchical management violation.

group export

The `mccli group export` command exports group settings (that is, name, domain, encryption method, and so forth), and assigned policy objects (dataset, schedule, and retention policy) to an XML file. However, exported groups exported do not contain a list of client members.

Syntax

```
mccli group export --name=String --file=String
[--domain=String(/)] [--force=Boolean(false)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--file=`*String*
Specifies the name of the export XML file to create. This argument is required.

`--force=`*Boolean(false)*
If `true`, then the option forces the overwrite of an existing XML file.

`--name=`*String*
Specifies which group to export. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

Event codes

22234	Group does not exist.

group move-client

The `mccli group move-client` command moves a client from one group to another.

Syntax

```
mccli group move-client --client-name=String
--old-group-name=String --name=String
[--client-domain=String(/)] [--domain=String(/)]
[--old-group-domain=String(/)]
```

Options

`--client-domain=`*String(/)*
Specifies the Avamar server domain that contains the client specified by `--client-name`.

`--client-name=`*String*
Specifies which client to move. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
Specifies the target group (that is, the group that receives the client following the move). This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--old-group-domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--old-group-name`.

`--old-group-name=`*String*
Specifies the source group (that is, the group which the client was a member of prior to the move). This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

Event codes

22234	Group does not exist.
22236	Client does not exist.
22241	Client is not a member of old group.
22242	Client is already a member of new group.
22243	Client group membership successfully updated.

group remove-client

The `mccli group remove-client` command removes a client from a group.

A client must always be a member of at least one group. Therefore, if the client belongs only to the group from which you want to remove it, then you must first add the client to another group using the `mccli group add-client` command.

Syntax

```
mccli group remove-client --client-name=String --name=String
[--client-domain=String(/)] [--domain=String(/)]
```

Options

`--client-domain=`*String(/)*
Specifies the Avamar server domain that contains the client specified by `--client-name`.

`--client-name=`*String*
Specifies which client to remove. This argument is required.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/clients/MyClient`), then the `--domain` argument is ignored.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
Specifies the group from which to remove the client. This argument is required.

Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

Event codes

22234	Group does not exist.
22236	Client does not exist.
22241	Client is not a member of old group.
22243	Client group membership successfully updated.
22246	Client is not a member of any other group.
22270	Unable to remove a client from a group.

group remove-proxy

The `mccli group remove-proxy` command removes the specified Avamar proxy from the specified group.

Syntax

```
mccli group remove-proxy --name=String
--proxy-name=String [--domain=String(/)]
[--proxy-domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
Specifies the group from which to remove the proxy. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--proxy-domain=`*String(/)*
Specifies the Avamar server domain that contains the proxy specified by `--proxy-name`.

`--proxy-name=`*String*
Specifies which proxy to remove. This argument is required.
Use `mccli client show` to return a list of valid proxy names.
If you supply a fully qualified prpxy name (for example, `/clients/MyProxy`), then the `--domain` argument is ignored.

Event codes

22234	Group does not exist.
22236	Client does not exist.
24001	Failed to update proxy client mappings of a group.
24004	Proxy client mappings of a group successfully updated.

Examples

This command removes proxy backupproxy225 from the Test1 group:

mccli group remove-proxy --domain=/vcenter-1.example.com --name=Test1 --proxy-domain=/clients --proxy-name=backupproxy225

```
0,24002,Proxy client mappings of a group successfully updated.
Attribute Value
--------- --------------------------
group     /vcenter-1.example.com/Test1
```

group replicate

The `mccli group replicate` command replicates the specified group. It is equivalent to the Avamar Administrator **Replicate Now** command.

Syntax

```
mccli group replicate --name=String
[--domain=String(/)] [--wait[=min]]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
: Specifies which group to replicate. This argument is required.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--wait`[=*min*]
: Specifies a time period in minutes (min) that this command waits for the initiated operation to complete. If the operation completes before then, status is shown sooner. This option also shows more detailed status. If `--wait=0` or `--wait` (no value) is supplied, the command waits indefinitely for the activity to complete.

Event codes

22226	Group disabled.
22234	Group does not exist.
22441	Scheduled client replicate (Success).
22801	Login failure.
23028	Invalid group type.
30930	Expired Retention (Expiration).
30978	No proxy for replication.

group show

The `mccli group show` command returns information for all groups.

Syntax

```
mccli group show [--clients=Boolean(false)]
[--domain=String(/)] [--name=String]
[--proxy-clients=Boolean(false)] [--recursive=Boolean(false)]
[--verbose=Boolean(false)]
```

Options

`--clients=`*Boolean(false)*
: If `true`, then all clients that belong to the group are listed.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
: Specifies the group for which to show information. If not supplied, all groups in the domain are listed.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

`--proxy-clients=`*Boolean(false)*
: If `true`, then the command shows all proxies belonging to this group. This option is only valid fro vCenter groups.

`--recursive=`*Boolean(false)*
: If `true`, then the command shows all groups in the domain and any subdomains specified by the `--domain` argument..
If `false` or not supplied, then the command only shows groups in the domain. Groups in subdomains are not displayed.

`--verbose=`*Boolean(false)*
: If `true`, then detailed information is returned.
If `false` or not supplied, then only the group names are returned.

Event codes

22542	Domain does not exist.

Examples

This command returns a simple list of all groups in use:

`mccli group show`

```
0,23000,CLI command completed successfully.
command completed successfully
Group         Domain Type
------------- ------ ------
Default Group /      Normal
```

This command returns detailed information for a single group:

`mccli group show --name='Default Group'`

```
0,23000,CLI command completed successfully.
command completed successfully
Attribute         Value
----------------- -----------------
Name              Default Group
Domain            /
Type              Normal
Enabled           false
Dataset           Default Dataset
Schedule          Default Schedule
Retention Policy  Default Retention
Encryption Method High
```

This command returns all proxies belonging to the Default Virtual Machine Group in vCenter 10.31.183.85:

mccli group show --proxy-clients=true
--name=/10.31.183.85/Default Virtual Machine Group

```
0,23000,CLI command completed successfully.
Group                                          Proxy
---------------------------------------------- ------------------------
/10.31.183.85/Default Virtual Machine Group clients/localhost-proxy-1
```

group show-members

The `mccli group show-members` command returns a list of all members for a group.

Syntax

```
mccli group show-members [--domain=String(/)] [--name=String]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the group specified by `--name`.

`--name=`*String*
Specifies the group for which to show client members. If not supplied, all groups in the domain are listed.
Use `mccli group show` to return a list of valid group names.
If you supply a fully qualified group name (for example, `/MyDomain/MyGroup`), then the `--domain` argument is ignored.

Event codes

22234	Group does not exist.

Examples

This command returns a simple list of all clients in the Default Virtual Machine Group:

mccli group show-client-members
--name=/10.31.183.55/Default Virtual Machine Group --xml

```
<CLIOutput>
  <Results>
    <ReturnCode>0</ReturnCode>
    <EventCode>23000</EventCode>
    <EventSummary>CLI command completed successfully.</EventSummary>
  </Results>
  <Data>
    <Row>
      <Group>/10.31.183.55/Default Virtual Machine Group</Group>
      <GroupType>Default Virtual Machine Group</GroupType>
      <Client>/10.31.183.55/FO2</Client>
      <ClientType>Virtual Container</ClientType>
    </Row>
    <Row>
      <Group>/10.31.183.55/Default Virtual Machine Group</Group>
      <GroupType>Default Virtual Machine Group</GroupType>
      <Client>/10.31.183.55/HleDynamicClients/TEST1</Client>
      <ClientType>Virtual Machine</ClientType>
   </Row>
    <Row>
      <Group>/10.31.183.55/Default Virtual Machine Group</Group>
      <GroupType>Default Virtual Machine Group</GroupType>
      <Client>/10.31.183.55/APP1</Client>
      <ClientType>Virtual Container</ClientType>
    </Row>
    <Row>
```

```
      <Group>/10.31.183.55/Default Virtual Machine Group</Group>
      <GroupType>Default Virtual Machine Group</GroupType>
      <Client>/10.31.183.55/HleDynamicClients/ABCD1</Client>
      <ClientType>Virtual Machine</ClientType>
    </Row>
    <Row>
      <Group>/10.31.183.55/Default Virtual Machine Group</Group>
      <GroupType>Default Virtual Machine Group</GroupType>
      <Client>/10.31.183.55/HleDynamicClients/VA1</Client>
      <ClientType>Virtual Container</ClientType>
    </Row>
  </Data>
</CLIOutput>
```

group update-client

The `mccli group update-client` command is used to update the override dataset of a client in a group.

Syntax

```
mccli group update-client --client-name=String --name=String
[--client-domian=String] [--dataset=String] [--dataset-
domain=String(/)]
[--domain=String(/)]
```

Options

`--client-domain`*String*
: Specifies the client's domian.

`--client-name=`*String*
: Specifies the name of the client to add.

`--dataset=`*String*
: Specifies the name of the override dataset.

`--dataset-domain=`*String*
: Specifies the domain of the dataset.

`--domain=`*String*
: Specifies the group's domain.

`--name=`*String*
: Specifies the name of the group.

Event codes

22234	Group does not exist.
22236	Client does not exist.
22243	Client group membership successfully updated.
23003	Invalid dataset name specified on the CLI

Examples

This command updates the datatset for a client:

```
mccli group update-client --name=14680-group --domain=/14680-domain --
client-name=14680-client --client-domain=/14680-domain --
dataset=14680-dataset2 --dataset-domain=/14680-domain
```

```
0,22243,Client group membership successfully updated.
```

help

The `mccli help` resource shows online help for a resource, and then exits.

Syntax

```
mccli help {activity | agent | backup
| checkpoint | client | dataset | dd | domain
| dump | event | group | help | mcs
| plugin | retention | schedule | server | user
| vcenter | version | vmcache}
```

mcs

The `mccli mcs` resource is used to control various MCS functions.

mcs import

The `mccli mcs import` command accepts an XML file created by the `mccli group export` command and imports the group and its policy objects (that is, dataset, schedule, and retention policy) into the target Avamar server. The group name remains the same, but you can specify a new target domain.

Syntax

```
mccli mcs import --file=String
[--force=Boolean(false)] [--prefix=String]
[--target-domain=String]
```

Options

`--file=`*String*
: Specifies the name of the import XML file. This argument is required.

`--force=`*Boolean(false)*
: XML files created by the export command are digitally signed. If an XML file is edited or manually created, then digital signature verification fails during import.
If `true`, then the XML import is forced without validation of the digital signature.
If `false` or not supplied, then an error is returned, and the file is not imported.

`--prefix=`*String*
: Specifies a prefix *String* that is affixed to the beginning of the imported object names. This is done to avoid conflicts and distinguish imported objects from existing ones that might otherwise have similar names.

`--target-domain=`*String*
: Specifies the domain on the target server for the group and its policy objects (that is, dataset, schedule, and retention policy).
Original domain hierarchy for the imported objects is not preserved. In other words, all imported objects reside at the top-most level of the domain.
The target domain must already exist on the target server. If not, then the command returns an error.
To avoid conflicts, if a policy object with the same name already exists in the target domain, then _1, _2, _3 and so forth is appended to the name of the newly imported policy object.

Event codes

22540	Invalid name character.

mcs list

The `mccli mcs list` command accepts an XML file created by the group export command and lists all groups and group policy objects (that is, dataset, schedule, and retention policy) in that file.

Syntax

```
mccli mcs list --file=String
```

Options

`--file=`*String*
: Specifies the name of the import XML file. This argument is required.

mcs reboot-proxy

The `mccli mcs reboot-proxy` command reboots VMware proxies.

Syntax

```
mccli mcs reboot-proxy [--all=Boolean(false)] [--name=String]
```

Options

`--all=`*Boolean(false)*
: If `true`, then reboot all known Avamar proxies.

`--name=`*String*
: Specifies which proxy to reboot. *String* must be a fully qualified Avamar proxy name. Use `mccli client show` to return a list of valid proxy names.
Multiple `--name=` arguments are allowed on the same command line.

mcs resume-scheduler

The `mccli mcs resume-scheduler` command turns on the MCS scheduler so that scheduled operations occur at the scheduled times.

Syntax

```
mccli mcs resume-scheduler
```

Event codes

22308	Scheduler successfully resumed.
22310	Change of the scheduler status to suspended or resume failed.
22631	Server has reached the capacity health check limit.

mcs scheduler-status

The `mccli mcs scheduler-status` command returns the status of the MCS scheduler.

Syntax

```
mccli mcs scheduler-status
```

mcs stop

The `mccli mcs stop` command shuts down the MCS.

Syntax

```
mccli mcs stop
```

mcs suspend-scheduler

The `mccli mcs suspend-scheduler` command disables the MCS scheduler. When this occurs, no scheduled operations are performed until the scheduler is re-enabled using the `mccli mcs resume-scheduler` command.

Syntax

```
mccli mcs suspend-scheduler
```

Event codes

22309	Scheduler successfully suspended.
22310	Change of the scheduler status to suspended or resume failed.

mcs waitforflushcomplete

The `mccli mcs waitforflushcomplete` command causes the MCS to wait for any in-progress flush to complete.

Syntax

```
mccli mcs waitforflushcomplete
```

plugin

The `mccli plugin` resource is used to manage client plug-ins on the Avamar server.

plugin show

The `mccli plugin show` command shows summary properties for all client plug-ins.

Syntax

```
mccli plugin show
```

plugin update

The `mccli plugin update` command updates the Avamar server plug-in catalog.

Note

The `mccli plugin update` command is strictly reserved for EMC internal use only.

Syntax

```
mccli plugin update
```

Event codes

21010	Plugin catalog dynamically updated.

retention

The `mccli retention` resource is used to create and manage backup retention policies.

Advanced retention descriptors

All advanced retention options, which begin with `--adv-`, require a descriptor as a value.

The following table lists valid advanced retention descriptors.

Table 9 Advanced retention descriptors

Descriptor	Description	Examples
+*nnn*D	Specifies the number (*nnn*) of calendar days (D).	Specify +5D to retain five daily backups. Specify +13D to retain 13 daily backups.
+*nn*W	Specifies the number (*nn*) of calendar weeks (W).	Specify +1W to retain one week of daily or weekly backups. Specify +13W to retain 13 weeks of daily or weekly backups.
+*nn*M	Specifies the number (*nn*) of calendar months (M).	Specify +1M to retain one month of daily, weekly, or monthly backups. Specify +9M to retain nine months of daily, weekly, or monthly backups.
+*nn*Y	Specifies the number (*nn*) of calendar years (Y).	Specify +1Y to retain one year of daily, weekly, monthly, or yearly backups. Specify +5Y to retain five years of daily, weekly, monthly, or yearly backups.

retention add

The `mccli retention add` command creates a retention policy.

Syntax

```
mccli retention add --name=String
[--adv-daily=String(+1W)] [--adv-weekly=String(+1M)]
[--adv-monthly=String(+1Y)] [--adv-yearly=String(+1Y)]
[--basic={ NO_EXPIRATION| YYYY-MM-DD | +nn{ D | W | M | Y}]
[--domain=String(/)] [--override=Boolean(false)]
```

Options

`--adv-daily=`*String(+1W)*
: Specifies the daily advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--adv-weekly=`*String(+1M)*
: Specifies the weekly advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--adv-monthly=`*String(+1Y)*
: Specifies the monthly advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--adv-yearly=`*String(+1Y)*
: Specifies the yearly advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--basic={``NO_EXPIRATION`| *YYYY-MM-DD* | +*nn*{`D`|`W`|`M`|`Y`}]
: Specifies basic retention, which assigns a fixed expiration date in one of the following formats:

  - *YYYY-MM-DD* — specifies an explicit calendar date
  - +*nn*`{D | W | M | Y}` — specifies a duration from today (for example, `+4W` specifies four weeks from today)
  - `NO_EXPIRATION` — specifies that the backup is retained indefinitely (that is, it never expires)

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the retention policy specified by `--name`.

`--name=`*String*
: Specifies a name for the new retention policy. This argument is required.
Use `mccli retention show` to return a list of valid retention policy names.
If you supply a fully qualified retention policy name (for example, `/MyDomain/MyRetentionPolicy`), then the `--domain` argument is ignored.

`--override=`*Boolean(false)*
: If `true`, then basic retention settings are overridden in favor of advanced retention settings. This option only applies to scheduled backups.

Notes

If you do not specify options, then the new retention policy has a basic expiration setting of `NO_EXPIRATION` and the advanced options are set as follows:

- Retain one week of daily backups.
- Retain one month of weekly backups.

- Retain one year of monthly backups.
- Retain one year of yearly backups.

However, unless the `--override` option is set `true`, then these settings are not enabled. Basic retention settings are used instead of advanced retention settings.

Event codes

22216	Retention Policy created.
23005	Invalid Retention Policy specified on the CLI.
23011	Retention Policy already exists.

retention copy

The `mccli retention copy` command copies an existing retention policy, creating a new retention policy.

Syntax

```
mccli retention copy --name=String --new-name=String
[--domain=String(/)] [--new-domain=String(/)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the retention policy specified by `--name`.

`--name=`*String*
: Specifies which retention policy to copy. This argument is required.
Use `mccli retention show` to return a list of valid retention policy names.
If you supply a fully qualified retention policy name (for example, `/MyDomain/MyRetentionPolicy`), then the `--domain` argument is ignored.

`--new-domain=`*String(/)*
: Specifies the Avamar server domain that contains the retention policy specified by `--new-name`.

`--new-name=`*String*
: Specifies a name for the new retention policy. This argument is required.
If you supply a fully qualified retention policy name (for example, `/MyDomain/MyRetentionPolicy`), then the `--domain` argument is ignored.

Event codes

22216	Retention Policy created.
22289	Retention policy does not exist.

retention delete

The `mccli retention delete` command permanently deletes a retention policy from the Avamar server.

You cannot delete a retention policy if it is currently assigned to a client or group.

Syntax

```
mccli retention copy --name=String [--domain=String(/)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the retention policy specified by `--name`.

`--name=`*String*
Specifies which retention policy to delete. This argument is required.
Use `mccli retention show` to return a list of valid retention policy names.
If you supply a fully qualified retention policy name (for example, `/MyDomain/MyRetentionPolicy`), then the `--domain` argument is ignored.

Event codes

22218	Retention Policy deleted.
22289	Retention policy does not exist.

retention edit

The `mccli retention edit` command edits the properties for a retention policy.

There are no default settings for the `mccli retention edit` command. If you enter the command but do not explicitly supply options and values on the command line, then no changes are applied to the retention policy.

Syntax

```
mccli retention edit --name=String
[--adv-daily=String(+1W)] [--adv-weekly=String(+1M)]
[--adv-monthly=String(+1Y)] [--adv-yearly=String(+1Y)]
[--basic={ NO_EXPIRATION| YYYY-MM-DD | +nn{ D | W | M | Y}]
[--domain=String(/)] [--override=Boolean(false)]
```

Options

`--adv-daily=`*String(+1W)*
Specifies the daily advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--adv-weekly=`*String(+1M)*
Specifies the weekly advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--adv-monthly=`*String(+1Y)*
Specifies the monthly advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--adv-yearly=`*String(+1Y)*
Specifies the yearly advanced retention setting.
String must be +*nnn*`D` , +*nn*`W`, +*nn*`M` , or +*nn*`Y`.

`--basic={NO_EXPIRATION`| *YYYY-MM-DD* | +*nn*{`D`|`W`|`M`|`Y`}]
Specifies basic retention, which assigns a fixed expiration date in one of the following formats:

- *YYYY-MM-DD* — specifies an explicit calendar date
- +*nn*{`D | W | M | Y`} — specifies a duration from today (for example, `+4W` specifies four weeks from today)
- `NO_EXPIRATION` — specifies that the backup is retained indefinitely (that is, it never expires)

`--domain=`*String(/)*

Specifies the Avamar server domain that contains the retention policy specified by `--name`.

`--name=`*String*
Specifies which retention policy to modify. This argument is required.
Use `mccli retention show` to return a list of valid retention policy names.
If you supply a fully qualified retention policy name (for example, `/MyDomain/MyRetentionPolicy`), then the `--domain` argument is ignored.

`--override=`*Boolean(false)*
If `true`, then basic retention settings are overridden in favor of advanced retention settings. This option only applies to scheduled backups.

Event codes

22217	Retention Policy modified.
22289	Retention policy does not exist.
23005	Invalid Retention Policy specified on the CLI.

retention show

The `mccli retention show` command returns information for all retention policies in an Avamar domain.

Syntax

```
mccli retention copy [--domain=String(/)] [--name=String]
[--recursive=Boolean(false)] [--verbose=Boolean(false)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the retention policy specified by `--name`.

`--name=`*String*
Specifies a retention policy name. If not supplied, all retention policies are listed with summary information.
Use `mccli retention show` to return a list of valid retention policy names.
If you supply a fully qualified retention policy name (for example, `/MyDomain/MyRetentionPolicy`), then the `--domain` argument is ignored.

`--recursive=`*Boolean(false)*
If `true`, then the command shows all retention policies in the domain and any subdomains specified by the `--domain` argument.
If `false` or not supplied, then the command only shows retention policies in the specified domain. Subdomains are not examined.

`--verbose=`*Boolean(false)*
If `true`, then detailed retention policy information is returned.
If `false` or not supplied, then only retention policy names are returned.
If you also supply `--name`, then detailed information for that retention policy is returned.

Event codes

22289	Retention policy does not exist.
22542	Domain does not exist.

Examples

This command returns verbose information for all retention policies:

```
mccli retention show --verbose
```

```
0,23000,CLI command completed successfully.
Name                          Expiration Override
----------------------------- ---------- --------
Default Retention             90 Days    Yes
End User On Demand Retention 90 Days    No
Minimal Retention             90 Days    No
Monthly Retention             1 Months   No
Weekly Retention              1 Weeks    No
```

This command lists information for the Default Retention policy:

```
mccli retention show --name='Default Retention'
```

```
0,23000,CLI command completed successfully.
Attribute             Value
--------------------- -----------------
Name                  Default Retention
Basic Expiration Date  90 Days
Override              Yes
Keep days of daily     1
```

schedule

The `mccli schedule` resource is used to manage Avamar server schedules.

schedule add

The `mccli schedule add` command creates a new schedule.

Syntax

```
mccli schedule add --name=String
[--days={SU | M | TU | W | TH | F | SA}]
[--desc=String] [--duration=String] [--domain=String(/)]
[--nth-day={[1-28]  | last}] [--on-demand=Boolean(false)]
[--start=String] [--time=String] [--tz=String]
[--week={first | second |  third | fourth | last}]
```

Options

`--days={SU|M|TU|W|TH|F|SA}`
: Specifies which day of the week the schedule will run.
Separate multiple values with commas.

`--desc=`*String*
: Specifies a short text description of the schedule.

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the schedule specified by `--name`.

`--duration=`*String*
: Specifies a backup window duration.
String must be in the format of *HH:MM*.
If you do not supply `--duration`, then the default duration for a daily schedule is one hour and the default duration for a weekly schedule is eight hours.

`--name=`*String*
: Specifies a name for the new schedule. This argument is required.

Use `mccli schedule show` to return a list of valid schedule names.
If you supply a fully qualified schedule name (for example, `/MyDomain/MySchedule`), then the `--domain` argument is ignored.

`--nth-day={[1-28] | last}`
Creates a schedule that runs on the specified calendar day of each month. *String* must be a valid numerical calendar date between `1` and `28`, or `last`.

`--on-demand=`*Boolean(false)*
If `true`, then the new schedule is designated as an on-demand schedule, which also overrides other schedule type designations.
If `false` or not supplied, then .

`--start=`*String*
Specifies the earliest start time for the schedule. *String* must be in the format of *HH:MM*.
The `--start` option does not apply for daily schedules.
If you do not specify `--start` for a weekly schedule, then the default start time is 10 p.m. in the time zone in which the schedule was created.

`--time=`*String*
Creates a schedule that runs daily at the specified time of the day. *String* must be in the format of *HH:MM*.
Multiple `--time` options can be specified on the same command line.

`--tz=`*String*
Specifies the time zone for the schedule. If not supplied, then the local time zone is used.
Use `mccli schedule show-timezones` to return a list of valid time zone names.

`--week={first | second | third | fourth | last}`
Creates a schedule that runs on the specified week each month.

Notes

You must specify either `--days`, `--nth-day`, `--time`, or `--week` to indicate the schedule recurrence. Specify `--days` to create a weekly schedule, `--time` to create a daily schedule, and `--nth-day` or `--week` to create a monthly schedule.

If you specify `--duration`, then the default duration for a daily schedule is one hour and the default duration for a weekly schedule is eight hours.

The `--start` option does not apply for daily schedules.

If you do not specify the `--start` option for a weekly schedule, then the default start time is 10 p.m. in the time zone in which the schedule was created.

If you do not specify the `--tz` option, then the default time zone is the local time zone.

Event codes

22213	Schedule created.
22248	Schedule already exists.
22540	Invalid name character.
22542	Domain does not exist.

Examples

This command creates a new monthly schedule that runs on the second Friday of each month:

```
mccli schedule add --name=monthly_2nd_fri --week=second --days=Friday
```

This command creates a new monthly schedule that runs on the 28th calendar day of each month:

```
mccli schedule add --name=monthly_28th --nth-day=28
```

schedule copy

The `mccli schedule copy` command copies an existing schedule, creating a new schedule.

Syntax

```
mccli schedule copy --name=String --new-name=String
[--domain=String(/)] [--new-domain=String(/)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the schedule specified by `--name`.

`--name=`*String*
: Specifies which schedule to copy. This argument is required.
Use `mccli schedule show` to return a list of valid schedule names.
If you supply a fully qualified schedule name (for example, `/MyDomain/MySchedule`), then the `--domain` argument is ignored.

`--new-domain=`*String(/)*
: Specifies the Avamar server domain that contains the schedule specified by `--new-name`.

`--new-name=`*String*
: Specifies a name for the new schedule. This argument is required.
If you supply a fully qualified schedule name (for example, `/MyDomain/MySchedule`), then the `--domain` argument is ignored.

Event codes

22213	Schedule created.
22248	Schedule already exists.
22249	Schedule does not exist.

schedule delete

The `mccli schedule delete` command permanently deletes a schedule from the Avamar server.

You cannot delete a schedule if it is assigned to a group or event profile.

Syntax

```
mccli schedule delete --name=String [--domain=String(/)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain that contains the schedule specified by `--name`.

`--name=`*String*

Specifies which schedule to delete. This argument is required.
Use `mccli schedule show` to return a list of valid schedule names.
If you supply a fully qualified schedule name (for example, `/MyDomain/MySchedule`), then the `--domain` argument is ignored.

Event codes

22215	Schedule deleted.
22249	Schedule does not exist.
22277	Failed to delete a schedule because it is being used by a group or an event profile.
22530	Schedule is read-only.
22531	Unexpected exception occurred.

schedule edit

The `mccli schedule edit` command edits the properties for a schedule.

There are no default settings for the `mccli schedule edit` command. If you enter the command but do not explicitly supply options and values on the command line, then there are no changes to the schedule.

Syntax

```
mccli schedule edit --name=String
[--days={SU | M | TU | W | TH | F | SA}]
[--desc=String] [--duration=String] [--domain=String(/)]
[--nth-day={[1-28]  | last}] [--on-demand=Boolean(false)]
[--start=String] [--time=String] [--tz=String]
[--week={first | second |  third | fourth | last}]
```

Options

`--days={SU|M|TU|W|TH|F|SA}`
Specifies which day of the week the schedule will run.
Separate multiple values with commas.

`--desc=`*String*
Specifies a short text description of the schedule.

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the schedule specified by `--name`.

`--duration=`*String*
Specifies a backup window duration.
String must be in the format of *HH:MM*.
If you do not supply `--duration`, then the default duration for a daily schedule is one hour and the default duration for a weekly schedule is eight hours.

`--name=`*String*
Specifies which schedule to modify. This argument is required.
Use `mccli schedule show` to return a list of valid schedule names.
If you supply a fully qualified schedule name (for example, `/MyDomain/MySchedule`), then the `--domain` argument is ignored.

`--nth-day={[1-28]|last}`
Creates a schedule that runs on the specified calendar day of each month. *String* must be a valid numerical calendar date between `1` and `28`, or `last`.

`--on-demand=`*Boolean(false)*

If `true`, then the new schedule is designated as an on-demand schedule, which also overrides other schedule type designations.
If `false` or not supplied, then .

`--start=`*String*
Specifies the earliest start time for the schedule. *String* must be in the format of *HH:MM*.
The `--start` option does not apply for daily schedules.
If you do not specify `--start` for a weekly schedule, then the default start time is 10 p.m. in the time zone in which the schedule was created.

`--time=`*String*
Creates a schedule that runs daily at the specified time of the day. *String* must be in the format of *HH:MM*.
Multiple `--time` options can be specified on the same command line.

`--tz=`*String*
Specifies the time zone for the schedule. If not supplied, then the local time zone is used.
Use `mccli schedule show-timezones` to return a list of valid time zone names.

`--week={first|second|third|fourth|last}`
Creates a schedule that runs on the specified week each month.

Notes

You must specify either `--days`, `--nth-day`, `--time`, or `--week` to indicate the schedule recurrence. Specify `--days` to create a weekly schedule, `--time` to create a daily schedule, and `--nth-day` or `--week` to create a monthly schedule.

If you specify `--duration`, then the default duration for a daily schedule is one hour and the default duration for a weekly schedule is eight hours.

The `--start` option does not apply for daily schedules.

If you do not specify the `--start` option for a weekly schedule, then the default start time is 10 p.m. in the time zone in which the schedule was created.

If you do not specify the `--tz` option, then the default time zone is the local time zone.

Event codes

22214	Schedule modified.
22249	Schedule does not exist.

Examples

This command modifies a monthly schedule to run on the last calendar day of each month.

```
mccli schedule edit --name=monthly --nth-day=last
```

schedule show

The `mccli schedule show` command lists all schedules and detailed schedule information.

Syntax

```
mccli schedule show [--domain=String(/)] [--name=String]
[--recursive=Boolean(false)] [--verbose=Boolean(false)]
```

Options

`--domain=`*String(/)*
Specifies the Avamar server domain that contains the schedule specified by `--name`.

`--name=`*String*
Specifies which schedule to delete. This argument is required.
Use `mccli schedule show` to return a list of valid schedule names.
If you supply a fully qualified schedule name (for example, `/MyDomain/MySchedule`), then the `--domain` argument is ignored.

`--recursive=`*Boolean(false)*
If `true`, then the command shows all schedules in the domain and any subdomains specified by the `--domain` argument.
If `false` or not supplied, then the command only shows schedules in the specified domain. Subdomains are not examined.

`--verbose=`*Boolean(false)*
If `true`, then detailed schedule information is returned.
If `false` or not supplied, then only schedule names are returned.
If you also supply `--name`, then detailed information for that schedule is returned.

Event codes

22249	Schedule does not exist.
22542	Domain does not exist.

Examples

This command returns detailed information for the Default Schedule:

```
mccli schedule show --name='Default Schedule'
```

```
0,23000,CLI command completed successfully.
Attribute                       Value
------------------------------- ---------------------------------
Name                            Default Schedule
Domain                          /
ReadOnly                        true
Native Timezone                 America/Los_Angeles
Daylight Savings Adjustment (m) 60
Next Run Time                   2014-09-20 10:00 PM
Start Time                      10:00 PM Pacific Daylight Time
Backup Window Duration (hours)  8.0
Repeat                          Weekly
Days of Week                    Sun, Mon, Tue, Wed, Thu, Fri, Sat
Delay Start Until               N/A
End Policy                      No End Date
Description                     N/A
```

schedule show-timezones

The `mccli schedule show-timezones` command lists all valid time zones, which you can supply with other `mccli schedule` commands using the `--tz` option.

Syntax

```
mccli schedule show-timezones
```

server

The `mccli server` resource is used to monitor various Avamar server functions.

server show-prop

The `mccli server show-prop` command returns detailed properties for the Avamar server.

If you run `mccli server show-prop` without options, then a summary of server properties is shown.

Syntax

```
mccli server show-prop [--gc=Boolean(false)]
[--maintenance=Boolean(false)]
{[--module={module | ALL}] |
[--node={module.node | ALL} [--partition={partition | ALL}]]}
[--verbose=]
```

Options

`--gc=`*Boolean(false)*
: If `true`, then detailed garbage collection information is shown.

`--maintenance=`*Boolean(false)*
: If `true`, then detailed maintenance activity information is shown.
: If `false` or not supplied, then .

`--module={`*module* | `ALL}`
: Specifies an Avamar server module.
: *String* must be a valid Avamar server module designator. Module designators are typically single-digit integers beginning with zero. For example, 0, 1, 2, and so forth.
: A value of `ALL` returns status for all modules.
: The `--module` and `--node` options are mutually exclusive.

`--node={`*module.node* | `ALL}`
: Specifies an Avamar server node.
: *String* must be a valid Avamar server numeric node designator in the format of *module.node*. For example, 0.0 is typically the first storage node in a multi-node server.
: A value of `ALL` returns status for all nodes.
: The `--module` and `--node` options are mutually exclusive.

`--partition={`*partition* | `ALL}`
: Specifies an Avamar server data partition.
: *String* must be a valid Avamar server partition designator. Partition designators are typically single-digit integers beginning with zero. For example, 0, 1, 2, and so forth.
: A value of `ALL` returns status for all server data partitions.
: The `--partition` option requires `--node`.

`--verbose=`
: If supplied, then maximum information is returned..

Examples

This command returns a summary of server properties:

`mccli server show-prop`

```
0,23000,CLI command completed successfully.
Attribute                       Value
```

```
-------------------------------- -------------------------
State                            Node Offline
Active sessions                  0
Total capacity                   1.3 TB
Capacity used                    91 GB
Server utilization               0.7%
Bytes protected                  0.0 bytes
Bytes protected quota            Not configured
License expiration               Never
Time since Server initialization 17 days 21h:13m
Last checkpoint                  2014-10-01 04:00:28 PDT
Last validated checkpoint        2014-09-30 22:00:28 PDT
System Name                      avamar-1
System ID                        1216944475@00:19:B9:BA:14:EA
HFSAddr                          avamar-1.example.com
HFSPort                          27000
IP address                       10.6.248.196
Number of nodes                  6
Nodes Online                     5
Nodes Offline                    1
Nodes Read-only                  0
Nodes Timed-out                  0
```

This command returns information for server node 0.0 partition zero (0):

mccli server show-prop --node=0.0 --partition=0

```
0,23000,CLI command completed successfully.
Attribute                        Value
-------------------------------- -------------------------
Node ID                          0.0
State                            ONLINE
Runlevel                         fullaccess
Accessmode                       mhpu+0hpu+0hpu
Port                             26000
Dispatcher                       27000
Server uptime                    600 days 18h:08m
Server utilization               0.1%
Number of stripes                22
Server version                   7.1.0-nnn
Version                          v2.6.9-67.ELsmp
Node uptime                      600 days 00h:07m
Total capacity                   86.0 TB
Capacity used                    6.4 TB
Load average                     0.22
CPU %                            1.9%
Disk reads                       0.11/sec
Disk writes                      2.21/sec
Network reads                    2K/sec
Network writes                   2K/sec
IP address                       10.6.249.38
Mac address                      00:13:72:59:8F:AE
Number of partitions             4
```

This command returns detailed garbage collection information:

mccli server show-prop --gc=true

```
0,23000,CLI command completed successfully.
Attribute                        Value
-------------------------------- -------------------------
Status                           idle
Result                           OK
Start time                       2014-4-10 08:15:00 UTC
End time                         2014-4-10 08:15:17 UTC
Passes                           3
Bytes recovered                  81.9 KB
Chunks deleted                   16
Index stripes                    6
Index stripes processed          6
```

This command returns detailed maintenance activity information:

```
mccli server show-prop --maintenance=true
```

```
0,23000,CLI command completed successfully.
Attribute Value
--------- -----
Suspended No
```

server show-services

The `mccli server show-services` command returns information about Avamar server services.

Syntax

```
mccli server show-services [--service=String]
```

Options

`--service=`*String*
: If supplied, shows information for that specific service.
If not supplied, then all services are shown.

server show-util

The `mccli server show-util` command returns capacity utilization information from the Avamar server.

Syntax

```
mccli server show-util {[--module={module | ALL}]
| [--node={module.node | ALL}]}
```

If you run `mccli server show-util` without command options, then a summary of server capacity utilization is shown.

Options

`--module={`*module*`|` `ALL}`
: Specifies an Avamar server module.
String must be a valid Avamar server module designator. Module designators are typically single-digit integers beginning with zero. For example, 0, 1, 2, and so forth.
A value of `ALL` returns status for all modules.
The `--module` and `--node` options are mutually exclusive.

`--node={`*module.node*`|` `ALL}`
: Specifies an Avamar server node.
String must be a valid Avamar server numeric node designator in the format of *module.node*. For example, 0.0 is typically the first storage node in a multi-node server.
A value of `ALL` returns status for all nodes.
The `--module` and `--node` options are mutually exclusive.

server start-service

The `mccli server start-service` command starts a service that is currently stopped.

Syntax

```
mccli server start-service --service=String
```

Options

`--service=`*String*
: Which service to start. This argument is required.
Use `mccli server show-services` to return a list of valid service names.

server stop-service

The `mccli server stop-service` command stops a service that is currently running.

Syntax

```
mccli server stop-service --service=String
```

Options

`--service=`*String*
: Which service to stop. This argument is required.
Use `mccli server show-services` to return a list of valid service names.

user

The `mccli user` resource is used to manage backup user accounts.

user add

The `mccli user add` command creates a backup user account for a client or domain.

Syntax

```
mccli user add {--client-domain=String
| --client-name=String [--client-domain=String]}
--name=String --password=String --password-confirm=String
--role={ActivityOperator | Administrator | Auditor | Backup
| BackupOperator | BackupRestore | BackupRestoreOperator | Restore
| RestoreIgnoreFilePermissions | RestoreOperator}
[--authenticator=String(Axion)]
```

Options

`--authenticator=`*String(Axion)*
: Specifies the authentication system to use to grant the new user access to the Avamar server.
String must be either `Axion` or the name of another valid authentication system that has been configured for use with the Avamar server; *String* is case-insensitive.
Use `mccli user show-auth` to return a list of valid authentication system names.

`--client-domain=`*String*

Specifies the Avamar server domain for the new user.
If you supply only `--client-domain`, then the user is added to the domain.
If you supply both `--client-domain` and `--client-name`, then the user is added to that client. If you supply only --client-domain, then the user is added to the domain.
Use `mccli domain show` to return a list of valid domain names.

Note

You cannot add new user accounts to the `MC_RETIRED` domain or to any clients in that domain.

`--client-name=`*String*
Specifies which client gets the new user account.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/client/MyClient`), then the `--domain` argument is ignored.

`--name=`*String*

`--password=`*String*

`--password-confirm=`*String*
Specifies the user name and password for the new user account. These arguments are required.

`--role={ActivityOperator | Administrator | Auditor | Backup | BackupOperator | BackupRestore | BackupRestoreOperator | Restore | RestoreIgnoreFilePermissions | RestoreOperator}`
Specifies a role for the user. This argument is required.
You can only assign the `RestoreIgnoreFilePermissions` role if you use an external authentication system.

Event codes

22528	Failed to add user.
22529	User added.
22540	Invalid name character.
22542	Domain does not exist.
22546	Password mismatch.
22548	User already exists.
22549	Invalid privilege.
22550	Invalid authenticator.

Examples

This command adds a new user account, jsmith, with Administrator privileges to `/clients/MyDomain`:

```
mccli user add --name=jsmith --client-domain=/clients/MyDomain
--role=Administrator --password=XXXXXX --password-confirm=XXXXXX
```

```
0,22529,User added.
Attribute Value
--------- ----------------------------------------------------
user      <appuser authentication="Avamar Authentication System"
          folder="/clients/MyDomain" name="jsmith"
          privilege="Administrator"/>
```

This command adds a new user account, jsmith, with backup privileges to `/clients/MyClient`:

```
mccli user add --name=jsmith --client-domain=/clients --client-name=MyClient
--role=Administrator --password=XXXXXX --password-confirm=XXXXXX
```

```
0,22529,User added.
Attribute Value
---------
------------------------------------------------------------------
user      <appuser authentication="Avamar Authentication System"
          folder="/clients/MyClient" name="jsmith" privilege="Back up
Only"/>
```

user authenticate

The `mccli user authenticate` command verifies user authentication settings. This is useful for testing user names, passwords, and authentication system settings before creating a new user account.

Syntax

```
mccli user authenticate {--client-domain=String
| --client-name=String [--client-domain=String]}
--name=String --password=String
[--authenticator=String(Axion)]
```

Options

`--authenticator=`*String(Axion)*
: Specifies the authentication system associated with the user account.
String must be either `Axion` or the name of another valid authentication system that has been configured for use with the Avamar server; *String* is case-insensitive.
Use `mccli user show-auth` to return a list of valid authentication system names.

`--client-domain=`*String*
: Specifies the Avamar server domain associated with the client or user account.
If you supply only `--client-domain`, then the user is authenticated to that domain.
If you supply both `--client-domain` and `--client-name`, then the user is authenticated to that client. If you supply only `--client-domain`, then the user is authenticated to the domain.
Use `mccli domain show` to return a list of valid domain names.

`--client-name=`*String*
: Specifies the name of the client associated with the user account.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/client/MyClient`), then the `--domain` argument is ignored.

`--name=`*String*

`--password=`*String*
: Specifies a user name and password. These arguments are required.

Event codes

22348	Successful authenticate.
22349	Failed authenticate.

22542	Domain does not exist.
22550	Invalid authenticator.
22801	Login failure.

user delete

The `mccli user delete` command permanently deletes a user from the Avamar server.

Syntax

```
mccli user delete {--client-domain=String
| --client-name=String [--client-domain=String]}
--name=String [--authenticator=String(Axion)]
```

Options

`--authenticator=`*String(Axion)*
: Specifies the authentication system associated with the user account.
String must be either `Axion` or the name of another valid authentication system that has been configured for use with the Avamar server; *String* is case-insensitive.
Use `mccli user show-auth` to return a list of valid authentication system names.

`--client-domain=`*String*
: Specifies the Avamar server domain associated with the client or user account.
If you supply only `--client-domain`, then the user is deleted from the domain.
If you supply both `--client-domain` and `--client-name`, then the user is deleted from that client. If you supply only `--client-domain`, then the user is deleted from that to the domain.
Use `mccli domain show` to return a list of valid domain names.

`--client-name=`*String*
: Specifies the name of the client associated with the user account.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/client/MyClient`), then the `--domain` argument is ignored.

`--name=`*String*
: Specifies which user to delete. This argument is required.

Event codes

22522	Failed user delete.
22523	User deleted.
22540	Invalid name character.
22542	Domain does not exist.
22547	User does not exist.
22550	Invalid authenticator.
22600	Server inactive.

user edit

The `mccli user edit` command edits the properties for a user.

Syntax

```
mccli user edit --name=String {--client-domain=String
| --client-name=String [--password=String] [--password-
confirm=String] [--client-domain=String]}
[--authenticator=String(Axion)]
[--role={ActivityOperator | Administrator | Auditor | Backup
| BackupOperator | BackupRestore | BackupRestoreOperator | Restore
| RestoreIgnoreFilePermissions | RestoreOperator}]
```

Options

`--authenticator=`*String(Axion)*

Specifies the authentication system associated with the user account.
String must be either `Axion` or the name of another valid authentication system that has been configured for use with the Avamar server; *String* is case-insensitive.
Use `mccli user show-auth` to return a list of valid authentication system names.

`--client-domain=`*String*

Specifies the Avamar server domain associated with the client or user account.
If you supply only `--client-domain`, then the user is deleted from the domain.
If you supply both `--client-domain` and `--client-name`, then the user is deleted from that client. If you supply only `--client-domain`, then the user is deleted from that to the domain.
Use `mccli domain show` to return a list of valid domain names.

`--client-name=`*String*

Specifies the name of the client associated with the user account.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/client/MyClient`), then the `--domain` argument is ignored.

`--name=`*String*

`--password=`*String*

`--password-confirm=`*String*

Specifies the user name and password for the user account. The `--name=` argument is required.

`--role={ActivityOperator | Administrator | Auditor | Backup | BackupOperator | BackupRestore | BackupRestoreOperator | Restore | RestoreIgnoreFilePermissions | RestoreOperator}`

Specifies a role for the user.
You can only assign the `RestoreIgnoreFilePermissions` role if you use an external authentication system.

Event codes

22520	User updated.
22521	Failed user update.
22542	Domain does not exist.
22546	Password mismatch.
22547	User does not exist.

22549	Invalid privilege.
22550	Invalid authenticator.
22551	Can not edit reserved user.
23001	Missing required arguments.

user show

The `mccli user show` command displays properties for users on an access list for a client or domain.

Syntax

```
mccli user show --name=String {--client-domain=String
| --client-name=String [--client-domain=String]}
[--authenticator=String(Axion)] [--recursive=Boolean(false)]
```

Options

`--authenticator=`*String(Axion)*
: Specifies the authentication system associated with the user account.
String must be either `Axion` or the name of another valid authentication system that has been configured for use with the Avamar server; *String* is case-insensitive.
Use `mccli user show-auth` to return a list of valid authentication system names.

`--client-domain=`*String*
: Specifies the Avamar server domain associated with the client or user account.
If you supply only `--client-domain`, then the user is deleted from the domain.
If you supply both `--client-domain` and `--client-name`, then the user is deleted from that client. If you supply only `--client-domain`, then the user is deleted from that to the domain.
Use `mccli domain show` to return a list of valid domain names.

`--client-name=`*String*
: Specifies the name of the client associated with the user account.
Use `mccli client show` to return a list of valid client names.
If you supply a fully qualified client name (for example, `/client/MyClient`), then the `--domain` argument is ignored.

`--name=`*String*

`--password=`*String*

`--password-confirm=`*String*
: Specifies the user name and password for the user account. The `--name` argument is required.

`--recursive=`*Boolean(false)*
: If `true`, then the command shows all users in the domain and any subdomains specified by the `--domain` argument.
If `false` or not supplied, then the command only shows users in the specified domain. Subdomains are not examined.

Event codes

22542	Domain does not exist.
22547	User does not exist.

user show-auth

The `mccli user show-auth` command displays all authentication systems configured for use with the Avamar server.

Syntax

```
mccli user show-auth
```

vcenter browse

The `mccli vcenter browse` command is used to browse a vCenter for virtual machines, either by specifying a folder path, or ESX server or datacenter names. Both Host and Clusters, and VMs and Templates views are supported.

Syntax

```
mccli vcenter browse --name=String
[--container-path=String] [--datacenter=String]
[--esx-host=String] [--folder=String]
[--recursive=Boolean(false)]
[--type={DATACENTER | DATASTORE | NETWORK | VAPP | VM}]
[--vsphere-hosts-clusters-view=Boolean(false)]
```

Options

`--container-path=`*String*
: WhenSpecifies a container path within the specified datacenter. *String* can be a vApp container in the VMs and Templates view, or a resource pool in the Hosts and Clusters view.

`--datacenter=`*String*
: Specifies a fully qualified datacenter name in vCenter.
This option is only valid when browsing for virtual machines or vApps.

`--esx-host=`*String*
: Specifies a fully qualified ESX server hostname in a datacenter.
This option is only valid when browsing for virtual machines or vApps in the vSphere Hosts and Clusters view (that is, `--vsphere-hosts-clusters-view=true`).

`--folder=`*String*
: Specifies a folder path within the specified datacenter.
This option is only valid when browsing for virtual machines or vApps in the vSphere VMs and Templates view (that is, `--vsphere-hosts-clusters-view=false`).

`--name=`*String*
: Specifies the vCenter name. This argument is required.

`--recursive=`*Boolean(false)*
: If `true`, then command browse all levels of the specified folder path in a datacenter.
This option is only valid when browsing for virtual machines or vApps.

`--type={DATACENTER | DATASTORE | NETWORK | VAPP | VM}`
: Specifies the type of entity that should be searched for by this command.

- `DATACENTER`—Browse for datacenters. Returns datacenter names and locations.
- `DATASTORE`—Browse datastores. Returns detailed datastore information such as Datacenters, Name, Type, Accessible/Not, Hosts.
- `NETWORK`—Browse for virtual networks. Returns all available networks and their hosts.

- `VAPP`—Browse for vApps.
- `VM`—Browse virtual machines. Returns detailed virtual machine information such as Name, Guest OS, Server, Location, Template, Powered On/Off, Changed Block Tracking, and Is protected.

The default entity type is `VM`.

`--vsphere-hosts-clusters-view=`*Boolean(false)*

If `true`, then the vSphere Hosts and Clusters view is processed.

If `false` or not supplied, then the vSphere VMs and Templates view is processed.

This option is only valid when browsing for virtual machines or vApps.

Examples

This command recursively browses vCenter vcenter-1.example.com for virtual machines:

mccli vcenter browse --name=vcenter-1.example.com --recursive=true --xml

```
<CLIOutput>
 <Results>
  <ReturnCode>1</ReturnCode>
  <EventCode>23999</EventCode>
  <EventSummary />
 </Results>
 <Data>
  <Row>
   <Name>ACMCommunity</Name>
   <GuestOS>debian5Guest</GuestOS>
   <Server>10.31.183.7</Server>
   <Location>/VAAYU-DEV-WIN/vm/Test1/Test1/ACMCommunity</Location>
   <Template>No</Template>
   <PoweredOn>Yes</PoweredOn>
   <ChangedBlock>No</ChangedBlock>
   <Protected>Yes</Protected>
  </Row>
  <Row>
   <Name>RHEL564BUILDDOWNLOAD224</Name>
   <GuestOS>rhel5_64Guest</GuestOS>
   <Server>10.31.183.7</Server>
   <Location>
    /VAAYU-DEV-WIN/vm/Discovered virtualmachine/RHEL564BUILDDOWNLOAD224
   </Location>
   <Template>No</Template>
   <PoweredOn>Yes</PoweredOn>
   <ChangedBlock>No</ChangedBlock>
   <Protected>No</Protected>
  </Row>
 </Data>
</CLIOutput>
```

This command browses vcenter-1.example.com for datastores:

mccli vcenter browse --name=vcenter-1.example.com --type=DATASTORE --xml

```
Name     Type Accessible Hosts       Datacenter
-------- ---- ---------- ----------- ------------
Storage1 VMFS Yes        10.31.183.7 VAAYU-DEV-WIN
Storage2 VMFS Yes        10.31.183.7 VAAYU-DEV-WIN
```

This command browses vCenter vcenter-1.example.com for virtual machines in datacenter VAAYU-DEV-WIN Hosts and Clusters view:

mccli vcenter browse --name=vcenter-1.example.com --datacenter=VAAYU-DEV-WIN --esx-host=10.31.183.7 --vsphere-hosts-clusters-view=true --xml

```
<CLIOutput>
 <Results>
  <ReturnCode>1</ReturnCode>
  <EventCode>23999</EventCode>
  <EventSummary />
 </Results>
 <Data>
  <Row>
   <Name>ACMCommunity</Name>
   <GuestOS>debian5Guest</GuestOS>
   <Server>10.31.183.7</Server>
   <Location>
    /VAAYU-DEV-WIN/host/10.31.183.7/10.31.183.7/ACMCommunity
   </Location>
   <Template>No</Template>
   <PoweredOn>Yes</PoweredOn>
   <ChangedBlock>No</ChangedBlock>
   <Protected>Yes</Protected>
  </Row>
  <Row>
   <Name>RHEL564BUILDDOWNLOAD224</Name>
   <GuestOS>rhel5_64Guest</GuestOS>
   <Server>10.31.183.7</Server>
   <Location>
    /VAAYU-DEV-WIN/host/10.31.183.7/10.31.183.7/RHEL564BUILDDOWNLOAD224
   </Location>
   <Template>No</Template>
   <PoweredOn>Yes</PoweredOn>
   <ChangedBlock>No</ChangedBlock>
   <Protected>No</Protected>
  </Row>
 </Data>
```

This command recursively returns a list of virtual networks and the hosts that use them:

mccli vcenter browse --type=NETWORK --name=10.31.183.55

```
Network    Host
---------- ------------
VM Network 10.31.183.14
VM Network 10.31.183.17
VM Network 10.31.183.7
```

This command browses for vApps in datacenter DCF1/DCF2 in folders FO1/FO2:

mccli vcenter browse --type=VAPP --name=10.31.183.55
--datacenter=DCF1/DCF2 --folder=FO1/FO2 --recursive=true --xml

```
Listening for transport dt_socket at address: 8003
<CLIOutput>
 <Results>
  <ReturnCode>1</ReturnCode>
  <EventCode>23999</EventCode>
  <EventSummary />
 </Results>
 <Data>
  <Row>
   <Name>VA1</Name>
   <GuestOS>N/A</GuestOS>
   <Server>10.31.183.17</Server>
   <Location>/DCF1/DCF2/vm/FO1/FO2/VA1/</Location>
   <Template>No</Template>
   <PoweredOn>No</PoweredOn>
   <ChangedBlock>No</ChangedBlock>
   <Protected>No</Protected>
  </Row>
 </Data>
</CLIOutput>
```

version show

The `mccli version show` command returns the version of MCCLI currently installed, and then exits.

Syntax

```
mccli version show
```

vmcache

The `mccli vmcache` resource is used to used to assist with debugging possible vCenter data cache synchronization issues.

Note

The `mccli vmcache` resource is strictly reserved for EMC internal use only.

vmcache show

The `mccli vmcache show` command is used to examine cached vCenter data.

Note

The `mccli vmcache show` command is strictly reserved for EMC internal use only.

Syntax

```
mccli vmcache show --name=String [--domain=String(/)] [--
esxds=Boolean(false)], [--esxdsname=String], [--
verbose=Boolean(false)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain with the virtual machine or container entity specified by the `--name` argument. This argument is required.

`--esxds=`*Boolean(false)*
: If `true`, shows the ESX host/Datastore.

`--esxdsname=`*String*
: Specifies the ESX host or Datastore name to be shown.

`--name=`*String*
: Specifies the virtual machine or container entity for which to show cached vCenter data. This argument is required.

`--verbose=`*Boolean(false)*
: If `true`, shows verbose details. Not for `--esxds=true`.

Examples

This command shows cached vCenter data for a single virtual machine:

```
mccli vmcache show --domain=vcenter-1.example.com/ContainerClients
-name=my-vm
```

This command shows cached vCenter data for a container entity:

```
mccli vmcache show --domain=vcenter-1.example.com --name=my-container
```

vmcache sync

The `mccli vmcache sync` command is used to manually synchronize cached vCenter data.

Note

The `mccli vmcache sync` command is strictly reserved for EMC internal use only.

Syntax

```
mccli vmcache sync --name=String [--domain=String(/)] [--
showresult=Boolean(false)] [--recursive=Boolean(true)]
```

Options

`--domain=`*String(/)*
: Specifies the Avamar server domain with the virtual machine or container entity specified by the `--name` argument. This argument is required.

`--name=`*String*
: Specifies the virtual machine or container entity for which to show cached vCenter data. This argument is required.

`--recursive=`*Boolean(true)*
: If `false`, does not syncronize cache data recursively for the container or vCenter.

`--showresult=`*Boolean(false)*
: If `true`, shows the cache data after synchronization (not verbose).

Examples

This command synchronizes a single virtual machine's cached vCenter data:

```
mccli vmcache sync --name=my-vm
--domain=vcenter-1.example.com/ContainerClients
```

This command synchronizes a container entity's cached vCenter data:

```
mccli vmcache sync --name=my-container --domain=vcenter-1.example.com
```

APPENDIX A

Utilities and Configuration Files

This chapter includes the following topics:

avsetup_mccli

The `avsetup_mccli` utility configures the MCCLI.

By default the MCCLI RPM installs essential files to the locations in the following table.

Table 10 MCCLI essential files

Location	Essential files
`/usr/local/avamar/bin`	Binary and executables
`/usr/local/avamar/doc`	Documentation and report templates
`/usr/local/avamar/lib`	Resource libraries and initial configuration files

The RPM documentation provides instructions on installing the MCCLI application to a folder other than the default `/usr/local/avamar` location.

The default paths in the following table are used by the MCCLI during command invocation.

Table 11 MCCLI default paths

Default path	Description
`/usr/java/jre1.7.0_72` or `/usr/java/jre1.8`	Location of JRE installation
`/usr/local/avamar`	Location of MCCLI installation (this is also known as $AVAMAR_ROOT)
`~/.avamardata/var`	Location of user command invocation data and logs (this is also known as $USER_ROOT)

You can modify these path assignments at any time by rerunning the `avsetup_mccli` utility.

Additionally, `avsetup_mccli` also prompts you to specify values for all mccli global options. When you specify these values during the interactive `avsetup_mccli` session, it automatically updates the `mcclimcs.xml` default options file.

mccli.xml

The `mccli.xml` preferences file contains the parameters that you can edit for the MCCLI application.

The default version of `mccli.xml` is located in `$AVAMAR_ROOT/lib`. Each time the MCCLI application is run, `$USER_ROOT/.avamardata/var/mc/cli_data/prefs` is examined to determine if a working copy of `mccli.xml` is present. If `mccli.xml` is not present in `$USER_ROOT/.avamardata/var/mc/cli_data/prefs`, then the default copy of `mccli.xml` is copied to that location from `$AVAMAR_ROOT/lib`.

When any MCCLI command is invoked, `$USER_ROOT/.avamardata/var/mc/cli_data/prefs/mccli.xml` is read, and those settings are used for that command session.

The mccli.xml contains the parameters described in the following table.

Table 12 Parameters in mccli.xml

Parameter	Essential files
syntax_file	Stores the location of the mcclisyntax.xml file.
mcs_config_file	Stores the location of the mcclimcs.xml default options file.
invalid_name_characters	Do not change.
valid_input_characters	Do not change.
valid_passwd_characters	Do not change.
user_name_maximum_length	Do not change.
account_name_maximum_length	Do not change.
backup_label_maximum_length	Do not change.
validate_ascii_input_enabled	Do not change.
event_monitor_display_limit	Sets the maximum number of events to retrieve.
detail_server_stats	Specifies whether to display detailed statistics for the server. Default is false.
wait_refresh_interval	Do not change.

mcclimcs.xml

The mcclimcs.xml is an XML file that stores custom mccli command parameters and profile settings that are used when you invoke any mccli command.

Default command parameters

The mcclimcs.xml preferences file can be used to set a default value for any mccli command parameter. Any default values set in this file are used unless another value is explicitly supplied on the command line. Additionally, these default values are global, meaning that they are used by all profiles.

Profiles

Each profile is an element in the XML document and is distinguishable by the mcsprofile attribute, which identifies the name of the profile. Each profile contains a list of default options to use with the MCS specified for that profile.

You can designate one profile as the default profile. This default profile is used if no MCS information is specified on the command line global options. Otherwise, the profile name of the MCS is all that is required on the command line, and the remainder of the options are read from the configuration file.

One or all of the options can be specified on the command line to override entries in the mcclimcs.xml file.

Note

If the server hostname or data port assignment are changed for any reason (for example, after running the `resite` utility), or the user account name or password used to run `mccli` commands is changed for any reason, you must manually update the corresponding settings in the `mcclimcs.xml` preferences file to account for those changes.

Behavior

The default version of `mcclimcs.xml` is located in `$AVAMAR_ROOT/lib`. Each time the MCCLI application is run, `$USER_ROOT/.avamardata/var/mc/cli_data/prefs` is examined to determine if a working copy of `mcclimcs.xml` is present.

If `$USER_ROOT/.avamardata/var/mc/cli_data/prefs/mcclimcs.xml` is not present, then the default copy of `mcclimcs.xml` is copied to that location from `$AVAMAR_ROOT/lib`.

Example 1 Make mccli activity show only return active jobs

This setting constrains the `mccli activity show` command to only show active jobs, as if the `--active=true` option was supplied on the command line.

```
<Resource Name="activity">
 <Command Name="show">
  <Options>
   <Option Name="active" Value="true" />
  </Options>
 </Command>
</Resource>
```

Example 2 Make mccli client add set new client data port to 29123

These settings affect the `mccli client add` command so that any new client is enabled and its page data port is set to 29123 as if the `--enabled=true` and `--pageport=29123` options were supplied on the command line.

```
<Resource Name="client">
 <Command Name="add">
  <Options>
   <Option Name="enabled" Value="true" />
   <Option Name="pageport" Value="29123" />
  </Options>
 </Command>
</Resource>
```

Example 3 Add a new service account profile to the mcclimcs.xml file and encrypt the account password

To add a new service account to the mcclimcs.xml file, duplicate the existing MCS section and update the duplicated section as follows:

Example 3 Add a new service account profile to the mcclimcs.xml file and encrypt the account password (continued)

```
<MCS
       mcsprofile="local"
       mcsaddr="address"
       mcsport="port"
       mcsuserid="MCUser"
       mcspasswd="password"
  />
  <MCS
       mcsprofile="service"
       mcsaddr="address"
       mcsport="port"
       mcsuserid="ServiceUserName"
       mcspasswd="ServiceUserPassword"
  />
```

Encrypt the profile password by running the following command as root:

```
mccipher encrypt -c /usr/local/avamar/lib/mcclimcs.xml:service
```

Delete the existing /home/admin/.avamardata/var/mc/cli_data/prefs/mcclimcs.xml file and verify that the new profile works by running the following command:

```
mccli activity show --mcsprofile=service
```

GLOSSARY

A

Avamar Administrator A graphical management console software application that is used to remotely administer an Avamar system from a supported Windows or Linux client computer.

Avamar client A computer or workstation that runs Avamar software and accesses the Avamar server over a network connection. Avamar client software comprises a *client agent* and one or more *plug-ins*.

Avamar server The server component of the Avamar client/server system. Avamar server is a fault-tolerant, high-availability system that efficiently stores the backups from all protected clients. It also provides essential processes and services required for data restores, client access, and remote system administration. Avamar server runs as a distributed application across multiple networked storage nodes.

B

browse The process of viewing data that is available for backup on a client computer or restore from the Avamar server.

C

checkpoint A server backup taken for the express purpose of assisting with disaster recovery of the Avamar server.

client A computer or workstation that runs Avamar software and accesses the Avamar server over a network connection. Avamar client software consists of a client agent and one or more plug-ins.

client agent A platform-specific software process that runs on the client and communicates with the Management Console Server (MCS) and with any plug-ins installed on that client.

client registration The process of establishing an identity with the Avamar server. When Avamar recognizes the client, it assigns a unique client ID (CID), which it passes back to the client during *client activation*.

See also registration

D

Data Domain system Disk-based deduplication appliances and gateways that provide data protection and disaster recovery (DR) in the enterprise environment.

dataset A policy that defines a set of files, directories, and file systems for each supported platform that are included or excluded in backups across a group of clients. A dataset is a persistent and reusable Avamar policy that can be named and attached to multiple groups.

domain A feature in Avamar Administrator that is used to organize large numbers of clients into named areas of control and management.

G

group A level of organization in Avamar Administrator for one or more Avamar clients. All clients in an Avamar group use the same group policies, which include the *dataset*, *schedule*, and *retention policy*.

group policy The *dataset*, *schedule*, and *retention policy* for all clients in an Avamar group.

J

JRE Java Runtime Environment.

M

MCS Management console server. The server subsystem that provides centralized administration (scheduling, monitoring, and management) for the Avamar server. The MCS also runs the server-side processes used by *Avamar Administrator*.

P

plug-in Avamar client software that recognizes a particular kind of data resident on that client.

plug-in options Options that you specify during backup or restore to control backup or restore functionality.

policy A set of rules for client backups that can be named and applied to multiple groups. Groups have dataset, schedule, and retention policies.

R

registration The process of establishing an identity with the Avamar server. When Avamar recognizes the client, it assigns a unique client ID (CID), which it passes back to the client during *client activation*.

See also client registration

restore An operation that retrieves one or more file systems, directories, files, or data objects from a backup and writes the data to a designated location.

retention The time setting to automatically delete backups on an Avamar server. Retention can be set to permanent for backups that should not be deleted from an Avamar server. Retention is a persistent and reusable Avamar policy that can be named and attached to multiple groups.

roles A setting in Avamar Administrator that controls which operations each user can perform in the Avamar server. Roles are assigned on a user-by-user basis.

S

schedule The ability to control the frequency and the start and end time each day for backups of clients in a group. A schedule is a persistent and reusable Avamar policy that can be named and attached to multiple groups.