electric
CLOUD

ElectricAccelerator

cmtool Reference and Users Guide

version 6.2

Contents

Chapter 1: Introduction

cmtool: The ElectricAccelerator Command-line Tool

cmtool provides access to the Cluster Manager through a command-line interface instead of using the web interface. With cmtool, you can write Perl scripts to access Cluster Manager information or manage builds. Almost all ElectricAccelerator® operations and tasks can be implemented with cmtool—with the exception of a few reports that are generated only from the web interface.

cmtool is used primarily for build and agent management, including commands for build class management, agent testing, and adding comments automatically.

Logging In

If you use cmtool outside of a job, you *must* invoke the ***cmtool login*** command to log in to the server. After logging in, cmtool saves information about the login session for use in future cmtool invocations. If you run cmtool as part of an ElectricAccelerator job, you do not need to log in because `--cmtool` uses the login session (and credentials) for that job.

To log in to cmtool:

```
cmtool login <username> <password>
```

To specify a session file, use the `--sessionFile=<fileName>` option, so you can use the same session for subsequent cmtool invocations.

Using cmtool

An invocation of cmtool identifies the Cluster Manager to contact, using the `--server` command-line option, followed by a list of commands to execute. Certain commands may have optional or required arguments.

For example, the following invocation receives all build requests that ran fewer than 10 jobs and orders the list [that ran the build] by host name.

```
cmtool --server easerver getBuilds-filter "job_count <10" --order host_name
```

General syntax for cmtool command usage:

```
cmtool [optional global argument(s)] <command> <required arguments> [optional
arguments]
```

Return Codes

0 = success (the command was correct; if no data meets the criteria, return is still 0)
1 = failure (command was invalid)

cmtool Commands

Commands are grouped into common usage sections for your convenience. All cmtool commands are listed in alphabetical order within each group. Each command is shown with its description.

- Agent Management
- Build Management
- Cluster Management
- Reporting
- User Management

See the next chapter, cmtool Command Reference which begins with a grouped list of commands.

Click a command name to go to an expanded version of the command, including its required arguments, optional arguments, available fields (where applicable), and a usage example.

When you receive a "returned response"—most cmtool command responses are returned in XML format.

Another source of cmtool 'help' is its own online help, for example:

- `cmtool --help-commands` prints a list of all commands with a short description.
- `cmtool --help-fields` displays a list of fields for a command.

Examples for MySQL

Note that all database examples provided in this guide are specific to MySQL. If you use Oracle or MSSQL, use syntax that is appropriate for your respective database.

Global Arguments (Optional)

Global arguments supply general information quickly, including cmtool online help.

Note: Global arguments support using the "=" sign character.

Global Arguments	Description

`--help=[`*command*`]`
 	
`--help [`*command*`]`
	Prints this message and exits. If a command is specified, prints the help text for that command.

`--help-commands`	Prints the list of available commands with a short description.

`--help-fields=<`*command*`>`
 	
`--help-fields <`*command*`>`
	Displays a list of fields for a command—requires the *<command>* argument.

`--version`	Prints cmtool version number.

`--server=<`*hostname*`>`
 	
`--server <`*hostname*`>`
	ElectricAccelerator server address. Defaults to the ACCELERATOR_SERVER environment variable. If this variable does not exit, default is to the localhost.

`--port=<`*port*`>`
 	
`--port <`*port*`>`
	HTTP listener port on the ElectricAccelerator server. Defaults to port 8030.

Global Arguments	Description

`--securePort=<securePort>`
 	
`--securePort <securePort>`
	HTTPS listener port on the ElectricAccelerator server. Defaults to port 8031.

`--secure`	Uses HTTPS to communicate with the ElectricAccelerator server.

`--timeout=<s>`
 	
`--timeout <s>`
	cmtool waits for a response from the server for a specified amount of time. Timeout for server communication defaults to 180 seconds (3 minutes) if no other time is specified. After the timeout, cmtool exits but the server will continue to process the command.

`--output=<style>`
 	
`--output <style>`
	Set output style—default is to XML:
	
	'xml' for an XML document
	'csv' for comma separated values
	'simple' for no formatting
	'silent' for no output

`--fields=<list>`
 	
`--fields <list>`
	List is a comma separated list of fields to emit when using 'csv' or 'simple' output styles. Default is to *all* fields.

`--sessionFile=<path>`
 	
`--sessionFile <path>`
	Overrides the location where session information will be stored.

Using runAgentCmd

IMPORTANT: Exercise caution when using the `runAgentCmd` command. Electric Cloud recommends using this command for documented scenarios only or under the direction of Electric Cloud Technical Support.

The `runAgentCmd` command enables you to run agent commands against the cluster.

Use this format: `cmtool --cm=<cm> runAgentCmd "agent command to run"`)

where <cm> is the IP address or name of your Cluster Manager.

Some of the possible reasons for using `runAgentCmd` include:

- setting agent-side breakpoints (see the Using Breakpoints topic in online help)
- configuring agent log rotation (see the Installation and Configuration Guide)
- getting and setting agent and EFS debug levels (Knowledge Base article KB-00020)
- configuring the stalled job killer (Knowledge Base article KB-00031)
- troubleshooting builds that appear to hang (Knowledge Base article KB-00036)

Chapter 2: cmtool Command Reference

Click on a command to go to more information for that command.

Agent Management

Commands	Description
changeAgentsEnabled	Changes the agent enabled status of one or more agents.
createAgentComment	Creates a new agent comment.
createResource	Creates a new resource definition.
createResourceComment	Creates a new resource comment.
deleteAgentComment	Deletes an agent comment.
deleteAgents	Deletes an agent, including all dependent records.
deleteResource	Deletes a resource definition.
deleteResources	Deletes multiple resource definitions.
getBuildComments	Deletes a resource comment.
getAgentComments	Retrieves a list of related agent comments.
getAgentPerformance	Retrieves the performance log of one or more agents.
getAgents	Retrieves a list of agents.
getAgentStatus	Retrieves the state of one or more agents.
getLsfInformation	Retrieves current information about the LSF interface.
getLsfJobs	Retrieves information about all jobs submitted to LSF.
getResource	Finds a resource with full detail by the resource ID number.

Commands	Description
getResources	Retrieves a list of all resources.
modifyAgentComment	Modifies an agent comment.
modifyResource	Modifies a resource definition.
modifyResourceComment	Modifies a resource comment.
setAgentDebug	Sets the agent debug level.

Build Management

Commands	Description
createBuildClass	Creates a build class.
createBuildClassComment	Creates a new build class comment.
createBuildComment	Creates a new build comment.
deleteBuild	Deletes a build, including all dependent records.
deleteBuilds	Deletes a set of builds, including all dependent records.
deleteBuildClass	Deletes a build class, including all dependent records.
deleteBuildClasses	Deletes a set of build classes, including all dependent records.
deleteBuildClassComment	Deletes a build class comment.
deleteBuildComment	Deletes a build comment.
getBuild	Finds a build with full detail by the build's ID number.
getBuilds	Retrieves a list of builds.
getBuildComments	Retrieves a list of related build comments.
getBuildClass	Finds a build class with full detail by its ID number.
getBuildClasses	Retrieves a list of build classes with limited detail.
getBuildClassComments	Retrieves a list of related build class comments.
getBuildUserStats	Retrieves a list of user build statistics, grouped by user name, IP address, or host name.
modifyBuild	Modifies a build.

Commands	Description
modifyBuildClass	Modifies a build class.
modifyBuildClassComment	Modifies a build class comment.
modifyBuildComment	Modifies a build comment.
setDatabaseConfiguration	Modifies database configuration settings.
stopBuild	Stops a build.

Cluster Management

Commands	Description
createServerComment	Creates a new server comment.
deleteLicense	Deletes a license.
deleteMessage	Deletes a specific message, including all dependent records.
deleteMessages	Deletes a set of messages, including all dependent records.
deleteServerComment	Deletes a server comment.
exportData	Exports Cluster Manager data to a file.
getLicense	Retrieves information about one license.
getLicenses	Retrieves all license data.
getMessage	Retrieves a particular message.
getMessages	Retrieves a list of messages.
getResourceStats	Retrieves resource usage statistics.
getServer	Retrieves the server configuration.
getServerComments	Retrieves a list of related server comments.
getVersion	Retrieves server version information.
importData	Imports Cluster Manager data from a file.
importLicenseData	Imports one or more licenses.
logMessage	Creates a custom message on the Cluster Manager Messages page.
modifyServer	Modifies the server configuration.

Commands	Description
modifyServerComment	Modifies a server comment.
shutdownServer	Stops the server.
testAgents	Tests one or more agents.

Reporting

Commands	Description
createFilter	Creates a named filter for a specific table.
deleteFilter	Deletes a named filter for a specific table.
getCurrentServerLoad	Retrieves information about the current resource load.
getFilter	Retrieves a named filter for a specific table.
getFilters	Retrieves a list of saved filters for the current user.
modifyFilter	Updates a named filter for a specific table.

User Management

Commands	Description
addGroupMember	Adds a user name to the member list for a specific group.
changeOwnUser	Modifies the currently logged-in user.
createGroup	Creates a new local group.
createUser	Creates a new local user.
deleteGroup	Deletes a local group.
deleteUser	Deletes a local user.
getAccessEntries	Retrieves permissions for all users and groups that were granted server access.
getGroupMembers	Retrieves a list of users in a specific group.
getGroups	Finds all groups known to the server. If "local" is true, returns local groups only. Returns a list of groupInfo elements.

Commands	Description
getEffectivePermissions	Retrieves the permissions for the currently logged-in user.
getPermissions	Retrieves permissions for a particular user or group.
getUser	Finds a specific user known to the server.
getUsers	Finds all users known to the server. If "local" is true, returns local users only.
getUserSettings	Retrieves settings for the currently logged-in user.
login	Logs in to the client with the appropriate credentials.
logout	Logs out of the client session.
modifyGroup	Modifies a local group.
modifyUser	Modifies a local user.
removeGroupMember	Deletes a user from a specific group member list.
setBuildEndNotification	Enables/disables notification when builds of this class end for the currently logged-in user.
setPermissions	Creates or modifies permissions for a user or group. The permissions are a space-separated list of permission names.
setUserSettings	Updates settings for the currently logged-in user.

cmtool Commands in Full Detail (by Group)

Agent Management

changeAgentsEnabled	
Description	Changes the agent enabled status of one or more agents.
Required arguments	*<enabled>* - Values are true/false
Optional arguments	--agentId *<unique, internal number that can change; assigned by Cluster Manager>* --agentName*<name defined by the host where the agent resides [numbers and/or letters]>* --filter *<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getAgents command.>* **Note:** There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using --filter for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes.

changeAgentsEnabled	
	Note: If no agent name, agent ID, or filter is specified, all agents are changed.

Usage example	`cmtool changeAgentsEnabled false`
	
	Disables all agents in the cluster.
	
	`cmtool changeAgentsEnabled true --agentName linagent1`
	
	Enables the agent named "linagent1."
	
	`cmtool changeAgentsEnabled true --filter "agent_name LIKE 'winbuild1-%'"`
	
	This enables all agents with a name that begins with "winbuild1-".

createAgentComment	
Description	Creates a new agent comment.
Required arguments	*<text>* - The text of the item. Either `agentId` or `agentName` must be specified.
Optional arguments	`--agentId` *<unique, internal number that can change; assigned by Cluster Manager>* `--agentName` *<name defined by the host where the agent resides [numbers and/or letters]>*
Usage example	`cmtool createAgentComment --agentName linagent "Agent has been running great"` Creates a comment for an agent named "linagent".

createResource	
Description	Creates a new resource definition. After creating a resource, ensure the server is configured to support resource management. You can use the modifyServer command to enable resource management.
Required arguments	*<resourceName>* - This name is used on the eMake parameter: `--emake-resource`, and can be specified in a build class. It is used in the `ea_resource` table and also matches the resource requirement string for eMake. *<hostMasks>* - This is a quote-enclosed, semi-colon delimited list of host name masks, used to identify the list of hosts that support a resource. "*" is the wildcard character.
Optional arguments	`--description` *<a quote-enclosed text description for your reference only>*
Usage example	`cmtool createResource R29 "rs*; rt*" --description "rs or rt hosts"` Creates a new resource named R29 that only uses hosts whose names start with 'rs' or 'rt'.

createResourceComment	
Description	Creates a new resource comment.
Required arguments	*<resourceId>* - A unique number that identifies each resource. *<text>* - The text of the item.
Optional arguments	None
Usage example	`cmtool createResourceComment 2 "This resource identifies production servers"`

deleteAgentComment	
Description	Deletes an agent comment.
Required arguments	*<commentId>* - The unique key that identifies a comment. Use `getAgentComments` to get a list of `commentId` numbers. Either `agentId` or `agentName` must be specified.
Optional arguments	`--agentId` *<unique, internal number that can change; assigned by Cluster Manager>* `--agentName` *<name defined by the host where the agent resides [numbers and/or letters]>*
Usage example	`cmtool deleteAgentComment 1008 --agentId 14` Deletes comment ID 1008 from agent 14 (14 is the cluster manager internal ID for the agent). To find out what the appropriate comment ID is, use the getAgentComments command, which will list the comments attached to a particular agent.

deleteAgents	
Description	Deletes one or more agents, including all dependent records.
Required arguments	None
Optional arguments	`--agentId` *<unique, internal number that can change; assigned by Cluster Manager>* `--agentName` *<name defined by the host where the agent resides [numbers and/or letters]>* `--filter` *<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getAgents command.>*
Usage example	`cmtool deleteAgents --agentName winbuild1` Deletes agent "winbuild1" and all associated comments.

deleteResource	
Description	Deletes a resource definition.
Required arguments	*<resourceId>* - A unique number that identifies each resource. Use the `getResources` command to get a list of resource IDs.
Optional arguments	None
Usage example	`cmtool deleteResource 3` Deletes the resource definition with an ID of 3.

deleteResources	
Description	Deletes multiple resource definitions.
Required arguments	None
Optional arguments	`--filter`*<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getResources command.>*
Usage example	`cmtool deleteResources`

deleteResourceComment	
Description	Deletes a resource comment.
Required arguments	*<resourceId>* - A unique number that identifies each resource. *<commentId>* - The unique key that identifies a comment. Use the `getResourceComments` command to get a list of comment IDs.
Optional arguments	None
Usage example	`cmtool deleteResourceComment 3 49` Deletes comment 49 from resource 3.

getAgentComments	
Description	Retrieves a list of related agent comments, or a specific comment (by using the --commentId option).
Required arguments	None
Optional arguments	--agentId *<unique, internal number that can change; assigned by Cluster Manager>* --agentName *<name defined by the host where the agent resides [numbers and/or letters]>* --commentId *<unique key that identifies a comment>* **Note:** Either agentId or agentName must be specified.

Available fields	**Field output name**	**Description**
	commentId	The unique key that identifies a comment.
	createTime	The time when the item was created.
	lastModifiedBy	The user who last modified the item.
	modifyTime	The time when the item was last modified.
	text	The text of the item.

Usage example	cmtool getAgentComments --agentName ahost-3 Retrieves all comments for agent "ahost-3".

getAgentPerformance	
Description	Retrieves the performance log of one or more agents.
Required arguments	None
Optional arguments	--agentId *<unique, internal number that can change; assigned by Cluster Manager>* --agentName *<name defined by the host where the agent resides [numbers and/or letters]>* --agents *<this can be a list of agents whose performance you want to see>* --buildId *<this further restricts the returned agents to those running a specific build ID>* --status *<true/false> <choose active or inactive agents (ok or not ok)>* --enabled *<true/false> <choose enabled or disabled agents only>*

getAgentPerformance		
Available fields	**Field output name**	**Description**
	agentName	This is the name of the agent as it appears on the web page (product UI).
	result	This is the performance information of the agent.
Usage example	cmtool getAgentPerformance --agentName SOL1-1 Returns the performance log of the agent named "SOL1-1".	

getAgents			
Description	Retrieves a list of agents.		
Required arguments	None		
Optional arguments	--agentId *<unique, internal number that can change; assigned by Cluster Manager>* --agentName *<name defined by the host where the agent resides [numbers and/or letters]>* --filter *<SQL query used to limit the result set for a request. See the possible values below.>* **Note:** There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using --filter for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes. --maxResults *<maximum number of elements to run from a query>* --firstResult *<starting index for the query result set>* **Note:** --firstResult takes values beginning with "0". A negative value indicates a record starting from the end of the set, counting backwards, so "-1" is the last record, "-2" is the next to last, and so on. --order *<SQL order by clause, used to specify ordering for the query result set.>* --profile *<level of detail to return from a query. Possible values are details (all information) and info (reduced information set).>*		
Available fields	**Field output name**	**Description**	**SQL query name for** --filter **and** --order
	a2aPort	The agent to agent protocol communication port.	a2a_port
	agentId	A unique, internal number assigned to each agent by Cluster Manager; this number can change.	id

getAgents			
	agentName	A name defined by the host where the agent resides [numbers and/or letters].	agent_name
	agentVersion	The agent version string.	agent_version
	availableResults	This is a count of 'max' or 'first' results if --maxResults or --firstResult is specified.	N/A
	buildId	A unique number assigned by Cluster Manager for each build.	current_build_id
	buildName	The build name that is the expanded build class tag.	N/A
	consolePort	The agent console port.	console_port
	efsVersion	The EFS version string.	efs_version
	enabled	The flag indicating if an agent is enabled or not.	enabled
	errorCount	The number of internal agent errors.	error_count
	hostName	The name of the machine where Electric Make was invoked.	host_name
	inPenaltyBox	A flag indicating Electric Make had a recent problem with this agent.	N/A
	ipAddress	The agent IP address.	ip_address
	lastErrorTime	The last time the agent experienced an error.	last_error_time
	lastPingTime	The last time the agent was pinged to determine its status.	last_ping_time
	platform	The operating system being used or supported. If an OS is specified for a build class, builds from other operating systems cannot affiliate themselves with this class.	platform

getAgents			
	port	The agent protocol communication port.	port
	restartCount	The number of agent restarts.	restart_count
	status	The agent status. 1= OK, but anything else is an error code.	status
	statusDetail	If the last status update resulted in an error, it contains the error string (or the "OK" string if no error occurred).	status_detail
	webPort	The agent web server port.	web_port
Usage example	`cmtool getAgents --filter "agent_name like '%SOL%'"` Retrieves a list of all agents whose names start with "SOL".		

getAgentStatus	
Description	Retrieves the state of one or more agents. By default, only active agents are returned. (Use `--status false` to list inactive agents.)
Required arguments	None
Optional arguments	`--agentId` *<unique, internal number that can change; assigned by Cluster Manager>* `--agentName` *<name defined by the host where the agent resides [numbers and/or letters]>* `--agents` *<this can be a list of agents whose status you want to see>* `--buildId` *<this further restricts the returned agents to those running a specific build ID>* `--status` *<true/false> <choose active or inactive agents (ok or not ok)>* `--enabled` *<true/false> <choose enabled or disabled agents only>*
Available fields	**Field output name** **Description**

getAgentStatus		
	`agentName`	This is the name of the agent as it appears on the web page (product UI).
	`result`	This is the text string that describes the current state of the agent.
Usage example	`cmtool getAgentStatus --agentName SOL1-1` Returns the status of the agent named "SOL1-1".	

getLsfInformation		
Description	Retrieves current information about the LSF interface. **Note:** LSF must be enabled to retrieve information.	
Required arguments	None	
Optional arguments	None	
Available fields	**Field output name**	**Description**
	`clusterName`	The name of the LSF grid cluster.
	`lsfAvailable`	"1" if LSF is available to the Cluster Manager.
	`masterName`	The LSF Master Host name.
	`numPendingAgentJobs`	The number of LSF jobs submitted by Cluster Manager that are waiting to run.
	`numRunningAgentJobs`	The number of LSF jobs Cluster Manager submitted that are running now.
	`statusMessage`	A message.
Usage example	`cmtool --output csv --fields getLsfInformation` Retrieves a Boolean value to indicate whether LSF is available or not.	

getLsfJobs	
Description	Retrieves information about all jobs submitted to LSF.
Required arguments	None

getLsfJobs	
Optional arguments	None

Available fields	Field output name	Description
	agentHostName	The machine name where the agent is running.
	jobNumber	The job number referencing a batch job submitted to LSF.
	jobStatus	The current status of an LSF job.
	resourceRequest	A request to the resource manager for a particular type of agent.
	submitTime	The time the job was submitted to LSF.

Usage example	`cmtool getLsfJobs` Retrieves all LSF job information.	

getResource	
Description	Finds a resource with full detail by the resource ID number.
Required arguments	*<resourceId>* - A unique number that identifies each resource. Use `getResources` to retrieve a list of `resourceId`s.
Optional arguments	None

Available fields	Field output name	Description
	hostMasks	This is a semi-colon delimited list of host name masks, used to identify the list of hosts that support a resource. "*" is the wildcard character.
	resourceId	A unique number that identifies each resource.
	resourceName	This name is used on the eMake parameter: `--emake-resource`, and can be specified in a build class.

Usage example	`cmtool getResource 7` Retrieves a resource with the ID of "7".	

getResources			
Description	Retrieves a list of all resources.		
Required arguments	None		
Optional arguments	`--filter` *<SQL query used to limit the result set for a request. See the possible values below.>* **Note:** There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using `--filter` for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes. `--maxResults` *<the maximum number of elements to run from a query>* `--firstResult` *<the starting index for the query result set>* **Note:** `--firstResult` takes values beginning with "0". A negative value indicates a record starting from the end of the set, counting backwards, so "-1" is the last record, "-2" is the next to last, and so on. `--order` *<SQL order by clause, used to specify ordering for the query result set.>* `--profile` *<the level of detail to return from a query. Possible values are details (all information) and info (reduced information set).>*		
Available fields	**Field output name**	**Description**	**SQL query name for** `--filter` **and** `--order`
	availableResults	This is a count of 'max' or 'first' results if `--maxResults` or `--firstResult` is specified.	N/A
	hostMasks	This is a semi-colon delimited list of host name masks, used to identify the list of hosts that support a resource. "*" is the wildcard character.	host_masks
	resourceId	A unique number that identifies each resource.	id
	resourceName	This name is used on the eMake parameter: `--emake-resource`, and can be specified in a build class.	resource_name
Usage example	`cmtool getResources --order resource_name` Retrieves a list of resources ordered by the resource name.		

getResourceComments	
Description	Retrieves resource comments.
Required arguments	*<resourceId>* - A unique number that identifies each resource.
Optional arguments	`--commentId`
Usage example	`cmtool getResourceComments 29`

modifyAgentComment	
Description	Modifies an agent comment.
Required arguments	*<commentId>* - The unique key that identifies a comment. *<text>* - The text of the item.
Optional arguments	`--agentId` *<unique, internal number that can change; assigned by Cluster Manager>* `--agentName` *<name defined by the host where the agent resides [numbers and/or letters]>* **Note:** Either `agentId` or `agentName` must be specified.
Usage example	`cmtool modifyAgentComment 1037 "changed comment" --agentName SOL1-1` Changes comment number 1037 on agent SOL1-1 to "changed comment."

modifyResource	
Description	Modifies a resource definition.
Required argument	*<resourceId>* - A unique number that identifies each resource.
Optional arguments	`--hostMasks` *< a semi-colon delimited list of host name masks, used to identify the list of hosts that support a resource. "*" is the wildcard character.>* `--resourceName` *<the unique name of the resource>* `--description` *<a text description for your reference only>*
Usage example	`cmtool modifyResource 27 --hostMasks "SOL*; SRL*"` Sets the host masks for resource 27 to "SOL*; SRL*"

modifyResourceComment	
Description	Modifies a resource comment. Use `getResources` to retrieve a list of resource IDs.
Required arguments	*<resourceId>* - A unique number that identifies each resource. *<commentId>* - A unique key identifying a comment. Use `getResourceComments` to retrieve a list of comment IDs. *<text>* - The text of the item.
Optional arguments	None
Usage example	`cmtool modifyResourceComment 1 1015 "new xxx"` Changes comment 1015 for resource 1

setAgentDebug		
Description	Sets the agent debug level. [see `getAgentStatus`] This command sends a message to the agent(s) in real time; therefore, the agents must be up and connected to the Cluster Manager to have any effect.	
Required arguments	*<level>* - Possible options are *all,commands*, *environment*, *fileinfo*, *log*, *other*, *profile*, *registry*, *requests*, *state*, *test*, *usage*, and *nothing*.	
Optional arguments	`--agentId` *<unique, internal number that can change; assigned by Cluster Manager>* `--agentName` *<name defined by the host where the agent resides [numbers and/or letters]>* `--status` *<true | false> <to choose active or inactive agents (ok or not ok)>* `--buildId` *<this further restricts the returned agents to those running a specific build ID>* `--enabled` *<true | false> <to choose enabled or disabled agents only>* `--agents` *<host>*[:*<port>*[:*<agentKey>*]] *<to specify individual agents based on their host name and listening port>*	
Available fields	**Field output name**	**Description**
	agentName	The name of the configured agent
	result	The configuration result
Usage example	`cmtool setAgentDebug profile --agentName SOLAgent-1` Sets SOLAgent-1's debug level to "profile."	

Build Management

createBuildClass	
Description	Creates a build class.
Required arguments	*<buildClassName>* - A name assigned by the user for the build class
Optional arguments	`--tagDefinition` *<a format string that defines the resulting build name>* `--annotationLevels` *<a comma-separated list of values chosen from: basic, history, file, lookup, and waiting.>* `--maxAgents` *<maximum number of agents to request for this build>* `--minAgents` *<minimum number of agents required for this build to run>* `--platform` *<operating system being used/supported. If an OS is specified for a build class, builds from other operating systems cannot affiliate themselves with this class. Also, platform must be either Windows, Linux, or Solaris.>* `--priority` *<build priority level. Values can be Low, High, or the default Normal. When assigning resources, an optional priority boost value (number) can be selected to give a build class preference over other builds of the same priority level. Higher boost values correspond to greater preference. Priority value 120 is normal, 220 is high, 20 is low. Each value can be adjusted up or down by 1-10 to "boost" the priority.>* `--annoUpload` *<If set to true, the annotation file is uploaded to Cluster Manager. Values can be Y, N, y, n, yes, no, Yes, or No.>* `--resourceRequest` *<request to the resource manager for a particular type of agent.Value=the name of a pre-existing resource.>*
Usage example	`cmtool createBuildClass batch --minAgents 5 --maxAgents 12 --priority 35 --resourceRequest blades`

createBuildClassComment	
Description	Creates a new build class comment.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class. Use `getBuildClass` to retrieve a list of build class IDs. *<text>* - The text of the item.
Optional arguments	None
Usage example	`cmtool createBuildClassComment 7 "This build class is for QA builds."`

createBuildComment	
Description	Creates a new build comment.
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build. Use `getBuilds` to retrieve a list of build IDs. *<text>* - The text of the item.
Optional arguments	None
Usage example	`cmtool createBuildComment 1044 "This is our gold build for release 7.0"`

deleteBuild	
Description	Deletes a build, including all dependent records.
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build. Use `getBuilds` to retrieve a list of build IDs.
Optional arguments	None
Usage example	`cmtool deleteBuild 1037` Deletes a build with a build ID of 1037.

deleteBuildClass	
Description	Deletes a build class, including all dependent records.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class. Use `getBuildClasses` to retrieve a list of build class IDs.
Optional arguments	None
Usage example	`cmtool deleteBuildClass 7` Deletes a build class with the ID of 7.

deleteBuildClasses	
Description	Deletes a set of build classes, including all dependent records.
Required	None

deleteBuildClasses	
arguments	
Optional arguments	`--filter`*<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getBuildClasses command.>* **Note:** If no filter is provided, all build classes (except the default) will be deleted.
Usage example	`cmtool deleteBuildClasses --filter "max_agents >20"` This command deletes all build classes with more than 20 maximum agents.

deleteBuildClassComment	
Description	Deletes a build class comment.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class. *<commentId>* - The unique key that identifies a comment. Use `getBuildClassComments` to retrieve a list of build class comment IDs.
Optional arguments	None
Usage example	`cmtool deleteBuildClassComment 6 1018` This command deletes comment 1018 for build class 6.

deleteBuildComment	
Description	Deletes a build comment.
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build. *<commentId>* - The unique key that identifies a comment. Use `getBuildComments` to retrieve a list of comment IDs.
Optional arguments	None
Usage example	`cmtool deleteBuildComment 1037 1019` Deletes build comment 1019 for the 1037 build.

deleteBuilds	
Description	Deletes a set of builds, including all dependent records. It is important to remove build logs periodically so they do not fill up the Cluster Manager's available disk space. Uploaded annotation is also considered part of build logs, so remember to clean up build logs regularly if annotation is frequently uploaded to the Cluster Manager. You can also manage build logs using the Cluster Manager web interface. Select the Builds tab, and then create and run a "Builds by Date" filter to display the set of builds that you want to remove. Click **Delete Filtered Builds** to remove the build logs from disk and from the database.
Required arguments	If no argument is provided, all builds will be deleted.
Optional arguments	`--filter`*<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getBuilds command.>*
Usage example	```cmtool deleteBuilds --filter "start_time <date_sub(curdate( ), interval 20 day)"``` Deletes all builds more than 20 days old. **Note:** This example is valid for MySQL only. If you use Oracle or MSSQL, use syntax that is appropriate for your respective database.

getBuild			
Description	Finds a build with full detail by the build's ID number.		
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build. Use `getBuilds` to retrieve a list of build IDs.		
Optional arguments	None		
Available fields	**Field output names**		

	`allocatedAgents`
	`buildClassId`
	`buildClassName`
	`buildId`
	`buildLogDir`
	`buildName`
	`commandLine`
	`conflicts`
	`cwd`
	`duration`
		`effectiveAgentAlloc`
		`emakeVersion`
		`historyExists`
		`historyFile`
		`hostName`
		`ipAddress`
		`jobCount`
		`lastRequestTime`
		`maxAgents`
		`minAgents`
			`osUserName`
			`platform`
			`priority`
			`resourceRequest`
			`result`
			`requestedAgents`
			`startTime`
			`userLabel`
			`userName`
			`waitTime`

Usage example	```cmtool getBuild 1000```		

getBuilds	
Description	Retrieves a list of builds.
Required arguments	None
Optional arguments	`--filter` *<SQL query used to limit the result set for a request. See the possible values below.>* **Note:** There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using `--filter` for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes. `--maxResults` *<maximum number of elements to run from a query>* `--firstResult` *<starting index for the query result set>* **Note:**`--firstResult` takes values beginning with "0". A negative value indicates a record starting from the end of the set, counting backwards, so "-1" is the last record, "-2" is the next to last, and so on. `--order` *<SQL order by clause, used to specify ordering for the query result set>* `--profile` *<level of detail to return from a query. Possible values are details (all information) and info (reduced information set).>* **Note:** You must set `--profile details` in order to print fields that are part of the details category.

Available fields	Field output name	Description	SQL query name for `--filter` **and** `--order`
	`allocatedAgents`	The number of currently assigned agents for this build.	N/A
	`availableResults`	This is a count of 'max' or 'first' results if `--maxResults` or `--firstResult` is specified.	N/A
	`buildClassId`	A unique number assigned by Cluster Manager for each build class.	`build_class_id`
	`buildClassName`	A name assigned by the user for the build class.	`build_class_name`
	`buildId`	A unique number assigned by Cluster Manager for each build.	`id`
	`buildLogDir`	The directory containing uploaded build logs.	N/A

getBuilds			
	`buildName`	The build name that is the expanded build class tag.	`build_name`
	`commandLine`	The original command-line invocation of Electric Make.	`command_line`
	`conflicts`	The number of conflicts in the build.	`conflicts`
	`cwd`	The current working directory where Electric Make was invoked.	`cwd`
	`duration`	The number of milli-seconds the build has been running. **Note:**`duration` for running builds is always 0.	`duration`
	`effectiveAgentAlloc`	The effective agent allocation percentage. 100% means eMake had all the hosts it needed all the time, while a lesser percentage means eMake had the hosts it needed for that percent of time. **Note:**`effective_agent_ alloc` for running builds is always 0.	`effective_agent_ alloc`
	`emakeVersion`	The Electric Make version used for this build.	`emake_version`
	`historyExists`	True means the history file existed and was used by the build.	`history_exists`
	`historyFile`	The name of the Electric Make history file.	`history_file`
	`hostName`	The name of the machine where Electric Make was invoked.	`host_name`
	`ipAddress`	The IP address of the machine where Electric Make was invoked.	`ip_address`
	`jobCount`	The total number of jobs	`job_count`

getBuilds			
		that ran for the build. **Note:**`job_count` for running builds is always 0.	
	`lastRequestTime`	The last time Electric Make requested agents for this build.	N/A
	`maxAgents`	The maximum number of agents to request for this build.	`max_agents`
	`minAgents`	The minimum number of agents required for this build to run.	`min_agents`
	`osUserName`	The OS-level name for the user who started Electric Make.	`os_user_name`
	`platform`	The operating system being used/supported. If an OS is specified for a build class, builds from other operating systems cannot affiliate themselves with this class.	`platform`
	`priority`	The build priority level. When assigning resources, an optional priority boost value can be selected to give a build class preference over other builds of the same priority level. Higher boost values correspond to greater preference.	`priority`
	`resourceRequest`	A request to the resource manager for a particular type of agent.	`resource_request`
	`result`	The build result code. -1 means the build is still running, 0-254 are actual exit codes, 256 means the build timed out, and 257 means the build was stopped.	`result`

getBuilds			
	requestedAgents	The number of agents Electric Make requested.	N/A
	startTime	The time the build was started.	start_time
	userLabel	The user-supplied label (via the eMake command-line), attached to the build.	user_label
	userName	The unique name of the user.	user_name
	waitTime	The number of seconds Electric Make was stalled because it had to wait for agents. **Note:**wait_time for running builds is always 0.	wait_time
Usage example	cmtool --output simple --fields "startTime,buildName,userId, duration" getBuilds --filter "duration >10000" Returns the start time, build name, userid, and duration of all builds that ran more than 10 seconds.		

getBuildComments	
Description	Retrieves a list of related build comments.
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build.
Optional arguments	--commentId *<unique key that identifies a comment>*
Available fields	**Field output name** — **Description**

Field output name	Description
commentId	The unique key that identifies a comment.
createTime	The time when the item was created.
lastModifiedBy	The last time the agent experienced an error.
modifyTime	The time when the item was last modified.
text	The text of the item.

Usage example	cmtool getBuildComments 1000 --commentId 1039 Retrieves comment #1039 for build 1000.

getBuildClass	
Description	Finds a build class with full detail by its ID.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class. Use `getBuildClasses` to retrieve a list of build class IDs.
Optional arguments	None
Available fields (see `getBuildClasses` for field descriptions)	**Field output names**

	annotationLevels annoUpload buildClassId buildClassName	defaultClass maxAgents minAgents notifyOnBuildEnd	platform priority resourceRequest tagDefinition

Usage example	`cmtool getBuildClasses --filter "min_agents <5"` Retrieves a list of build classes that require less than 5 agents.

getBuildClasses			
Description	Retrieves a list of build classes with limited detail.		
Required arguments	None		
Optional arguments	`--filter` *<SQL query used to limit the result set for a request. See the possible values below.>* **Note:** There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using `--filter` for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes. `--maxResults` *<maximum number of elements to run from a query>* `--firstResult` *<starting index for the query result set>* **Note:**`--firstResult` takes values beginning with "0". A negative value indicates a record starting from the end of the set, counting backwards, so "-1" is the last record, "-2" is the next to last, and so on. `--order` *<SQL order by clause, used to specify ordering for the query result set.>* `--profile` *<level of detail to return from a query. Possible values are details (all information) and info (reduced information set).>*		
Available fields	**Field output name**	**Description**	**S**QL query name for `--filter` **and** `--order`
	annotationLevels	Annotation choices to include in the annotation	annotation_levels

getBuildClasses			
		file. Possible values are basic, history, file, lookup, and waiting.	
	`annoUpload`	If set to true, the annotation file is uploaded to Cluster Manager.	`anno_upload`
	`availableResults`	This is a count of 'max' or 'first' results if `--maxResults` or `--firstResult` is specified.	N/A
	`buildClassId`	A unique number assigned by Cluster Manager for each build class.	`id`
	`buildClassName`	A name assigned by the user for the build class.	`build_class_name`
	`defaultClass`	If set, this is the default build class and cannot be deleted.	`default_class`
	`maxAgents`	The maximum number of agents to request for this build.	`max_agents`
	`minAgents`	The minimum number of agents required for this build to run.	`min_agents`
	`notifyOnBuildEnd`	If set to true, the currently logged-in user will receive an email when the build is finished.	`notify_on_build_end`
	`platform`	The operating system being used/supported. If an OS is specified for a build class, builds from other operating systems cannot affiliate themselves with this class.	`platform`
	`priority`	The build priority level. When assigning resources, an optional priority boost value can be selected to give a build class preference over other builds of the same priority	`priority`

getBuildClasses			
		level. Higher boost values correspond to greater preference.	
	resourceRequest	A request to the resource manager for a particular type of agent.	resource_request
	tagDefinition	A format string that defines the resulting build name.	tag_definition
Usage example	cmtool getBuildClass 1		

getBuildClassComments	
Description	Retrieves a list of related build class comments.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class. Use getBuildClasses to retrieve a list of build class IDs.
Optional arguments	--commentId *<unique key that identifies a comment>*
Available fields	**Field output name** / **Description**

Available fields	Field output name	Description
	commentId	The unique key that identifies a comment.
	createTime	The time when the item was created.
	lastModifiedBy	The last time the agent experienced an error.
	modifyTime	The time when the item was last modified.
	text	The text of the item.

Usage example	cmtool getBuildClassComments 12 Retrieves all build class comments for build class 12.

getBuildUserStats	
Description	Retrieves a list of user build statistics, grouped by user name, IP address, or host name.
Required arguments	*<groupBy>* - Possible values are hostName, ipAddress, and userName.
Optional arguments	--filter *<SQL query used to limit the result set for a request. See the possible values below.>*

getBuildUserStats			
	Note: There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using `--filter` for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes.		
	`--order` *<SQL order by clause, used to specify ordering for the query result set.>*		
Available fields	**Field output name**	**Description**	**SQL query name for** `--filter` **and** `--order`
	`duration`	The total number of milli-seconds of all builds, filtered by the value specified in the `groupBy` argument.	`duration`
	`entryName`	The value specified in the `groupBy` argument. If `groupBy` is "userName", the entry name is the user name.	
	`numOfBuilds`	The number of builds.	
	`waitTime`	The number of seconds Electric Make was stalled because it had to wait for agents.	`wait_time`
	`workload`	The total number of seconds used by the agents for all of the filtered builds.	`workload`
Usage example	`cmtool getBuildUserStats hostName --filter "duration >30000"` `--order "waitTime desc, entryName asc"` Retrieves build user statistics for builds longer than 30 seconds, grouped by host name and ordered by wait time in a descending order and by entry name (in this case host name) in an ascending order.		

modifyBuild	
Description	Modifies a build.
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build. *<priority>* - The build priority level. Value can be Low or Normal, but NOT High.

modifyBuild	
Optional arguments	None
Usage example	`cmtool modifyBuild 1137 20` Changes build 1137 to priority 20.

modifyBuildClass	
Description	Modifies a build class.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class.
Optional arguments	`--buildClassName` *<name assigned by the user for the build class>* `--tagDefinition` *<format string that defines the resulting build name>* `--annotationLevels` *<a comma-separated list of possible values are: basic, history, file, lookup, and waiting>* `--maxAgents` *<maximum number of agents to request for this build>* `--minAgents` *<minimum number of agents required for this build to run>* `--platform` *<the operating system being used/supported. If an OS is specified for a build class, builds from other operating systems cannot affiliate themselves with this class. The value can be Windows, Linux, or Solaris.>* `--priority` *<the build priority level. When assigning resources, an optional priority boost value can be selected to give a build class preference over other builds of the same priority level. Higher boost values correspond to greater preference. Value can be Low, High, or the default Normal.>* `--annoUpload` *<if set to true, the annotation file is uploaded to Cluster Manager. Values can be Y, N, y, n, Yes, No, yes, no.>* `--resourceRequest` *<request to the resource manager for a particular type of agent. Value is the name of the pre-existing resource.>*
Usage example	`cmtool modifyBuildClass 1 --annoupload true` Changes build class 1 to upload annotation files.

modifyBuildComment	
Description	Modifies a build comment.
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build. *<commentId>* - The unique key that identifies a comment. *<text>* - The text of the item.

modifyBuildComment	
Optional arguments	None
Usage example	`cmtool modifyBuildComment 16975 1137 "This is not a usable build"`

modifyBuildClassComment	
Description	Modifies a build class comment.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class. *<commentId>* - The unique key that identifies a comment. *<text>* - The text of the item.
Optional arguments	None
Usage example	`cmtool modifyBuildClassComment 1037 1129 "This is a low-priority class"`

setDatabaseConfiguration	
Description	Modifies database configuration settings
Required arguments	*<databaseName>* - The database instance name. *<databaseType>* - The database type. Possible values are mysql, oracle, and sqlserver. *<hostName>* - Machine name where the database is installed. *<port>* - Database port number. *<userName>* - Unique name of the user that is used to access the database. *<password>* - Secret value used to identify an account for a particular user.
Optional arguments	None

stopBuild	
Description	Stops a running build. (This command has no effect on completed builds.)
Required arguments	*<buildId>* - A unique number assigned by Cluster Manager for each build. Use `getBuilds --filter "result <0"` to retrieve a list of running builds.

stopBuild	
Optional arguments	None
Usage example	`cmtool stopBuild 16937`

Cluster Management

createServerComment	
Description	Creates a new server comment. Server comments are displayed on the Home page of the Cluster Manager machine.
Required arguments	*<text>* - The text of the item.
Optional arguments	None
Usage example	`cmtool createServerComment "cluster needs more servers to handle production builds"`

deleteLicense	
Description	Deletes a license.
Required arguments	*<productName>* - ElectricAccelerator, the name of the license. *<featureName>* - Feature name of the license, which is currently the "server".
Optional arguments	None
Usage example	`cmtool deleteLicense ElectricAccelerator Server` Deletes the license stored in the server.

deleteMessage	
Description	Deletes a specific message, including all dependent records. Messages are listed in the Cluster Manager interface Messages tab and generally are notifications about issues with agents or the Cluster Manager.
Required arguments	*<messageId>* - The numeric value that uniquely identifies each message.

deleteMessage	
Optional arguments	None
Usage example	`cmtool deleteMessage 501`

deleteMessages	
Description	Deletes a set of messages, including all dependent records.
Required arguments	None
Optional arguments	`--filter` *<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getMessages command.>*
Usage example	`cmtool deleteMessages --filter "create_time <date_sub(curdate( ), interval 200 day)"` Removes all messages more than 200 days old. **Note:** This example is valid for MySQL only. If you use Oracle or MSSQL, use syntax that is appropriate for your respective database.

deleteServerComment	
Description	Deletes a server comment.
Required arguments	*<commentId>* - The unique key that identifies a comment.
Optional arguments	None
Usage example	`cmtool deleteServerComment 1396`

exportData	
Description	Exports Cluster Manager data to a file.
Required arguments	*<fileName>* - The filename or path to export to. If you use a filename, the destination is the current working directory of the Java process, for example, /opt/ecloud/i686_Linux or C:\ECloud\i686_win32. If you use a path, the Cluster Manager Java user must have execute and write access to the destination path.

exportData	
	Note: This is a full database dump and it may take an extended period of time to complete depending on the size of the database.
Optional arguments	None
Usage example	`cmtool exportData fileabc`

getLicense	
Description	Retrieves information for one license.
Required arguments	*\<productName\>* - ElectricAccelerator, the name of the license. *\<featureName\>* - Feature name of the license.
Optional arguments	None
Usage example	`cmtool getLicense ElectricAccelerator Server`

getLicenses	
Description	Retrieves all license data.
Required arguments	None
Optional arguments	None
Usage example	`cmtool getLicenses`

getMessage	
Description	Retrieves a particular message.
Required arguments	*\<messageId\>* - The numeric value that uniquely identifies each message.
Optional arguments	None

getMessage	
Available fields (see `getMessage-s` for field name descriptions)	**Field output name**

	`agentId`
	`agentName`
	`buildId`
	`buildName` `createTime`
	`messageId`
	`severity`
	`text`

Usage example	`cmtool --output csv --fields buildId,severity,text getMessages --filter`
	 `"text like '%I/O%'"`
	
	Lists all messages in the Cluster Manager that contain the string 'I/O'.

getMessages			
Description	Retrieves a list of messages		
Required arguments	None		
Optional arguments	`--filter` *<SQL query used to limit the result set for a request. See the possible values below.>* **Note:** There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using `--filter` for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes. `--maxResults` *<maximum number of elements to run from a query>* `--firstResult` *<starting index for the query result set>* **Note:**`--firstResult` takes values beginning with "0". A negative value indicates a record starting from the end of the set, counting backwards, so "-1" is the last record, "-2" is the next to last, and so on. `--order` *<SQL order by clause, used to specify ordering for the query result set.>* `--profile` *<level of detail to return from a query. Possible values are details (all information) and info (reduced information set).>*		
Available fields	**Field output name**	**Description**	**SQL query name for** `--filter` **and** `--order`
	`agentId`	A unique, internal number assigned to each agent by Cluster Manager; this number can change.	N/A
	`agentName`	A name defined by the host where the agent resides [numbers and/or letters].	`agent_name`

getMessages			
	buildId	A unique number assigned by Cluster Manager for each build.	build_id
	buildName	The build name that is the expanded build class tag.	N/A
	createTime	The time when the item was created.	create_time
	messageId	The numeric value that uniquely identifies each message.	id
	severity	The severity level of the event: Info, Warning, or Error. For --filter and --order, use the following numerical values: 1 = Info 2 = Warning 3 = Error	severity
	text	The text of the item.	text
Usage example	cmtool getMessage 47		

getResourceStats	
Description	Retrieves resource usage statistics.
Required arguments	None
Optional arguments	--filter *<SQL query used to limit the result set for a request. See the possible values below.>* **Note:** There is a syntax difference between MySQL and Oracle/MSSQL for enclosing criteria when using --filter for specific strings—for MySQL, use double quotes; for Oracle/MSSQL, use single quotes. --maxResults *<maximum number of elements to run from a query>* --firstResult *<starting index for the query result set>* **Note:**--firstResult takes values beginning with "0". A negative value indicates a record starting from the end of the set, counting backwards, so "-1" is the last record, "-2" is the next to last, and so on. --order *<SQL order by clause, used to specify ordering for the query result set.>*

getResourceStats			
	`--profile` *<level of detail to return from a query. Possible values are details (all information) and info (reduced information set).>*		
Available fields	**Field output name**	**Description**	**SQL query name for** `--filter` **and** `--order`
	`agentClusterShortage`	The difference between the maximum number of agents requested by all builds and the number of agents that were assigned.	`agent_cluster_ shortage`
	`agentDemand`	The total maximum number of requests for agents by all running builds.	`agent_demand`
	`agentLicenseShortage`	The difference between the maximum request for agents by all builds and the number of agents the license allows.	`agent_license_ shortage`
	`agentsAvailable`	The total number of active agents in the cluster.	`agents_available`
	`agentsInUse`	The total number of agents assigned to builds.	`agents_in_use`
	`availableResults`	This is a count of 'max' or 'first' results if `--maxResults` or `--firstResult` is specified.	N/A
	`buildsDuration`	The average amount of time the current builds have been running.	`builds_duration`
	`buildsRunning`	Average number of simultaneous builds running during a specific time period.	`builds_running`
	`createTime`	The time when the item was created.	`create_time`
	`duration`	The number of milli-seconds the build has been running.	`duration`
	`resourceName`	This name is used on the eMake parameter: `--`	`resource_name`

getResourceStats			
		emake-resource, and can be specified in a build class. It is used in the ea_ resource table and also matches the resource requirement string for eMake.	
	resourceStatId	The resource ID number that uniquely identifies every resource.	id
Usage example	cmtool getResourceStats --maxResults 100 --order "id desc" --filter "resource_name='Cluster'" Retrieves the 100 most current resource statistic records for the entire cluster.		

getServer	
Description	Retrieves server configuration.
Required arguments	None
Optional arguments	None

getServer		
Available fields	**Field output name**	**Description**
	`agentAllocationPolicy`	Defined as either *exclusive* or *shared*.
	`agentLockTimerSec`	When jobs run beyond this number of seconds, the agent should be locked.
	`badAgents`	The number of enabled agents with a bad status.
	`disabledAgents`	The number of disabled agents.
	`emailInterval`	The number of minutes between email notifications.
	`emailItemLimit`	Maximum number of messages per email notification.
	`goodAgents`	The number of enabled agents with a good status.
	`logDaysToKeep`	The number of days to keep message log entries.
	`lsfAvailable`	True if LSF is available to the Cluster Manager.
	`mailFrom`	The value to use in the From header element.
	`mailPrefix`	The string used to prefix subject lines.
	`maxAgents`	The maximum number of agents to request for this build.
	`maxClockSkew`	Specify the maximum clock skew (in seconds) allowed between the Electric Make client and agents in the cluster.
	`minAgents`	The minimum number of agents required for this build to run.
	`preemptionPolicy`	The allocation preemption policy.
	`priority`	The build priority level. When assigning resources, an optional priority boost value can be selected to give a build class preference over other builds of the same priority level. Higher boost values correspond to greater preference.

getServer		
	resourceManagerType	The type of resource manager that Cluster Manager should employ.
	resourceStatInterval	In minutes, the interval to collect stats on resource usage.
	resourceStatKeep	The number of minutes of resource usage statistics to keep.
	runningBuilds	The number of incomplete builds in the system.
Usage example	cmtool getServer	

getServerComments		
Description	Retrieves a list of related server comments.	
Required arguments	None	
Optional arguments	--commentId *<unique key that identifies a comment>*	
Available fields	**Field output name**	**Description**
	commentId	The unique key that identifies a comment.
	createTime	The time when the item was created.
	lastModifiedBy	The last time the agent experienced an error.
	modifyTime	The time when the item was last modified.
	text	The text of the item.
Usage example	cmtool getServerComments Returns all comments related to the server.	

getVersion	
Description	Retrieves server version information.
Required arguments	None
Optional arguments	None

getVersion		
Available fields	**Field output name**	**Description**
	`label`	The Electric Cloud build label for the server.
	`protocolVersion`	The server protocol version.
	`schemaVersion`	The server database schema version.
	`version`	The string identifying a component version.
Usage example	`cmtool getVersion`	

importData	
Description	Imports Cluster Manager data from a file.
Required arguments	*<fileName>* - The name of the file to import. The file's path is relative to the current working directory of the Java process, for example, /opt/ecloud/i686_Linux or C:\ECloud\i686_win32. **Note:** Because this imports a full database dump, be advised of the following: - The import may take an extended period of time to complete depending on the size of the database. - You must manually delete any old/unused agents from the agents list. - You must update the license file after import if it previously expired.
Optional arguments	None
Usage example	`cmtool importData fileabc`

importLicenseData	
Description	Imports one or more licenses.
Required arguments	*<licenseFile>* - Name of the file containing the license with the path.
Optional arguments	None
Usage example	`cmtool importLicenseData ./license.xml`

logMessage	
Description	Creates a custom message on the Cluster Manager Messages page.
Required arguments	*"message text"*
Optional arguments	`--severity` *<severity>* - Possible values are: Debug, Info, Warning, Error (or 0, 1, 2, 3). `--buildId` *<buildId>* - The message applies to the specified build only. `--agentName` *<agentName>* - The message applies to the specified agent name only. **Note:** If `--buildId` and `--agentName` are on the same line, the messasge is applied to build and the agent name.
Usage example	`cmtool logMessage "some text"`

modifyServer	
Description	Modifies the server configuration.
Required arguments	None

Optional arguments	`--priority` *<The default build priority level. When assigning resources, an optional priority boost value can be selected to give a build class preference over other builds of the same priority level. Higher boost values correspond to greater preference.>*
	
	`--emailInterval` *<number of minutes between email notifications>*
	
	`--emailItemLimit` *<maximum number of messages per email notification>*
	
	`--agentAllocationPolicy` *<exclusive or shared>*
	
	`--preemptionPolicy` *<the allocation preemption policy>*
	
	`--maxClockSkew` *<specify the maximum clock skew (in seconds) allowed between the Electric Make client and agents in the cluster>*
	
	`--maxAgents` *<maximum number of agents to request for this build>*
	
	`--minAgents` *<minimum number of agents required for this build to run>*
	
	`--resourceManagerType` *<values= none, ea, lsf, cloud, prioritypool to define what Cluster Manager should employ>*
	
	`--mailFrom` *<value to use in the From header element>*
	
	`--mailPrefix` *<string used to prefix subject lines>*
	
	`--logDaysToKeep` *<the number of days to keep message log entries>*
	
	`--resourceStatInterval`
	
	`--resourceStatKeep`
	
	`--wideDeepAllocationPolicy`*<deep or wide - Deep means the agent allocation*

modifyServer	
	algorithm favors assigning more agents on the same host to a build. Wide means the algorithm favors assigning more agents from different hosts. If wide, be sure `--agentAllocationPolicy` *is set to shared.>*
Usage example	`cmtool modifyServer --mailFrom "cm@ourhost.com" --mailPrefix "cm message:"` Changes the mail "from" and mail prefix values used for mail notifications sent by the server.

modifyServerComment	
Description	Modifies a server comment.
Required arguments	*<commentId>* - The unique key that identifies a comment. *<text>* - The text of the item.
Optional arguments	None
Usage example	`cmtool modifyServerComment 1178 "Server is fine"`

shutdownServer	
Description	Stops the server. **Note:** Use with caution, it works!
Required arguments	None
Optional arguments	`--restart` *<true | false>*
Usage example	`cmtool shutdownServer`

testAgents	
Description	Instructs the Cluster Manager to contact each active agent and update its status.
Required arguments	None

testAgents	
Optional arguments	`--agentId`*<unique, internal number that can change; assigned by Cluster Manager>* `--agentName` *<name defined by the host where the agent resides [numbers and/or letters]>* `--filter` *<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getAgents command.>*
Usage example	`cmtool testAgents --filter "agent_name like '%bl%'"` This command contacts all agents whose name contains 'bl' and updates their status.

Reporting

createFilter	
Description	Creates a named filter for a specific table. **Note:** Non-global filters are stored by user ID; therefore, the same name can be used by more than one user.
Required arguments	*<tableName>* - A short string that uniquely identifies the table being filtered. **Note:** Possible table names are: ec_agent, ec_build, ec_build_class, ec_filter, ec_ message, ec_resource, ec_resource_stat. *<filterName>* - A short string that uniquely identifies the filter. *<filterQuery>* - A SQL order by clause for the associated table.
Optional arguments	`--order` *<SQL order by clause, used to specify ordering for the query result set>* `--global` *<if true, this is a globally visible filter>*

Usage example	`cmtool createFilter ec_agents linuxAgents ""platform = 'linux'"`
	`--global true`
	
	Creates a global filter that selects only Linux agents.

deleteFilter	
Description	Deletes a named filter for a specific table.
Required arguments	*<tableName>* - A short string that uniquely identifies the table being filtered. **Note:** Possible table names are: ec_agent, ec_build, ec_build_class, ec_filter, ec_ message, ec_resource, ec_resource_stat. *<filterName>* - A short string that uniquely identifies the filter.

deleteFilter	
Optional arguments	`--global` *<if true, this is a globally visible filter>*
Usage example	`cmtool deleteFilter ec_agents linuxAgents --global true`

getCurrentServerLoad		
Description	Retrieves information about the current resource load.	
Required arguments	None	
Optional arguments	None	
Available fields	**Field output name**	**Description**
	`agentsAvailable`	The total number of active agents in the cluster.
	`agentClusterShortage`	The difference between the maximum number of agents requested by all builds and the number of agents that were assigned.
	`agentDemand`	The total maximum number of requests for agents by all running builds.
	`agentLicenseShortage`	The difference between the maximum request for agents by all builds and the number of agents the license allows.
	`agentsInUse`	The total number of agents assigned to builds.
	`buildsDuration`	The average amount of time the current builds have been running.
	`buildsRunning`	Average number of simultaneous builds running during a specific time period.
	`createTime`	The time when the item was created.
	`duration`	The number of milli-seconds the build has been running.

getCurrentServerLoad		
	resourceName	This name is used on the eMake parameter: `--emake-resource`, and can be specified in a build class. It is used in the `ea_resource` table and also matches the resource requirement string for eMake.
	resourceStatId	The resource ID number that uniquely identifies every resource.
Usage example	`cmtool getCurrentServerLoad`	

getFilter	
Description	Retrieves a named filter for a specific table.
Required arguments	*<tableName>* - A short string that uniquely identifies the table being filtered.
	Note: Possible table names are: ec_agent, ec_build, ec_build_class, ec_filter, ec_message, ec_resource, ec_resource_stat.
	<filterName> - A short string that uniquely identifies the filter.
Optional arguments	`--global` *<if true, this is a globally visible filter>*
Usage example	`cmtool getFilter ec_agent agentFilter`

getFilters	
Description	Retrieves a list of saved filters for the current user.
Required arguments	None
Optional arguments	`--filter`*<SQL query used to limit the result set for a request. For a list of possible SQL values, see the getAgents command.>*
	`--maxResults` *<maximum number of elements to run from a query>*
	`--firstResult` *<starting index for the query result set>*
	Note:`--firstResult` takes values beginning with "0". A negative value indicates a record starting from the end of the set, counting backwards, so "-1" is the last record, "-2" is the next to last, and so on.
	---order *<SQL order by clause, used to specify ordering for the query result set>*
Usage example	`cmtool getFilters --filter "table_name = 'ec_agent' && user_name is null"`
	Retrieves a list of all global filters for the agent table.

modifyFilter	
Description	Updates a named filter for a specific table.
Required arguments	*<tableName>* - A short string that uniquely identifies the table being filtered. **Note:** Possible table names are: ec_agent, ec_build, ec_build_class, ec_filter, ec_message, ec_resource, ec_resource_stat. *<filterName>* - A short string that uniquely identifies the filter. *<filterQuery>* - A SQL order by clause for the associated table.
Optional arguments	`--order` *<SQL order by clause, used to specify ordering for the query result set>* `--global` *<if true, this is a globally visible filter. This parameter is required for global filters.>*
Usage example	`cmtool modifyFilter ec_agent agentFilter "id 750" --order agent_name`

User Management

addGroupMember	
Description	Adds a user name to the member list for a specific group.
Required arguments	*<groupName>* - The unique name of the group. *<userName>* - The unique name of the user.
Optional arguments	None
Usage example	`cmtool addGroupMember DevGroupA ec123` Adds user 'ec123' to group DevGroupA.

changeOwnUser	
Description	Modifies the settings for the currently logged-in user.
Required arguments	*<userName>* - The unique name of the user.
Optional arguments	`--fullUserName` *<real world name of the user>* `--email` *<the associated user email address>* `--password` *<secret value used to identify an account for a particular user>* `--passwordFile` *<path to password file, if `--password` is also specified, `--`*

changeOwnUser	
	passwordFile *overrides its value in the command line>*
Usage example	cmtool ec123 --fullUserName "Mary Smith"

createGroup	
Description	Creates a new local group.
Required arguments	*<groupName>* - The unique name of the group.
Optional arguments	None
Usage example	cmtool createGroup DevGroupA

createUser	
Description	Creates a new local user.
Required arguments	*<userName>* - The unique name of the user. *<password>* - The secret value used to identify an account for a particular user.
Optional arguments	--fullUserName *<real world name of the user>* --email *<the associated user email address>* --passwordFile *<path to password file, if* --password *is also specified,* --passwordFile *overrides its value in the command line>*
Usage example	cmtool createUser ec123 psword --fullUserName "Bob Smith" --email "ec123@ourhost.com" Creates a new user named "ec123" whose real-world name is Bob Smith; with "psword" as his password. **Note:** If you do not wish to expose passwords on the command line, you can omit the password from the example above. Press the Enter key after typing the command string (without the password) and you will be prompted for the password.

deleteGroup	
Description	Deletes a local group.
Required	*<groupName>* - The unique name of the group.

deleteGroup	
arguments	
Optional arguments	None
Usage example	`cmtool deleteGroup DevGroupA` Removes the 'DevGroupA' group from the Cluster Manager.

deleteUser	
Description	Deletes a local user.
Required arguments	*<userName>* - The unique name of the user.
Optional arguments	None
Usage example	`cmtool deleteUser ec123`

getAccessEntries		
Description	Retrieves permissions for all users and groups that were granted server access.	
Required arguments	None	
Optional arguments	None	
Available fields	**Field output name**	**Description**
	`entityName`	A user or group name in an access entry.
	`permissions`	The list of permission flags for a particular entity.
Usage example	`cmtool getAccessEntries`	

getGroupMembers	
Description	Retrieves a list of users in a specific group.
Required arguments	*<groupName>* - The unique name of the group.
Optional arguments	None
Available fields	**Field output name**	**Description**
	`userName`	The unique name of the user.
Usage example	`cmtool getGroupMembers` Retrieves a list of user name elements.

getGroups	
Description	Finds all groups known to the server. If "local" is true, returns local groups only.
Required arguments	*<userName>* - The unique name of the user.
Optional arguments	`--local` *<if true, returns local users only>*
Available fields	**Field output name**	**Description**
	`groupName`	The unique name of the group.
	`mutable`	True if the associated user or group record is modifiable.
	`providerName`	The human-readable name configured for the directory provider of a specific user or group.
Usage example	`cmtool getGroups` Returns a list of `groupInfo` elements.

getEffectivePermissions	
Description	Retrieves the permissions for the currently logged-in user.
Required arguments	None
Optional arguments	None

getEffectivePermissions		
Available field	**Field output name**	**Description**
	`permissions`	The list of permission flags for a particular entity
Available permission flags	**Permission names**	

	`AgentsDelete`
	`AgentsRead`
	`AgentsWrite`
	`BuildsDelete`
	`BuildsRead`
	`BuildsWrite`
	`ClassesDelete`
	`ClassesRead`
	`ClassesWrite`
		`EMakeImpersonate`
		`EMakeInvoke`
		`MaintenanceDelete`
		`MaintenanceRead`
		`MaintenanceWrite`
		`MessageLogDelete`
		`MessageLogRead`
		`MessageLogWrite` `ReportsDelete`
		`ReportsRead`
		`ReportsWrite`
		`ResourcesDelete`
		`ResourcesRead`
		`ResourcesWrite`
		`ServerAccess`
		`UserModify`

Usage example	`cmtool getEffectivePermissions` Retrieves the permissions for the currently logged-in user.	

getPermissions		
Description	Retrieves permissions for a particular user or group.	
Required arguments	*<principalType>* - Value = user or group. *<entityName>* - A user or group name in an access entry.	
Optional arguments	None	
Available field	**Field output name**	**Description**
	`permissions`	The list of permission flags for a particular entity
Available permission flags	**Permission names**	

	`AgentsDelete`
	`AgentsRead`
	`AgentsWrite`
	`BuildsDelete`
	`BuildsRead`
	`BuildsWrite`
	`ClassesDelete`
	`ClassesRead`
	`ClassesWrite`
		`EMakeImpersonate`
		`EMakeInvoke`
		`MaintenanceDelete`
		`MaintenanceRead`
		`MaintenanceWrite`
		`MessageLogDelete`
		`MessageLogRead`
		`MessageLogWrite` `ReportsDelete`
		`ReportsRead`
		`ReportsWrite`
		`ResourcesDelete`
		`ResourcesRead`
		`ResourcesWrite`
		`ServerAccess`
		`UserModify`

Usage example	`cmtool getPermissions group DevGroupA` Retrieves permissions for group DevGroupA.	

getUser	
Description	Finds a specific user known to the server.
Required arguments	*<userName>* - The unique name of the user.
Optional arguments	None

Available fields	**Field output name**	**Description**
	email	The associated user email address.
	fullUserName	The real world name of the user.
	groupName	The unique name of the group.
	mutable	True if the associated user or group record is modifiable.
	providerName	The human-readable name configured for the directory provider of a specific user or group.
	userName	The unique name of the user.

Usage example	`cmtool getUser ec123` Retrieves the attributes for user ec123.

getUsers		
Description	Finds all users known to the server. If "local" is true, returns local users only.	
Required arguments	None	
Optional arguments	`--pattern` *<a wildcard pattern for a user name where "*" matches any character or SQL "like" string. If LDAP is set up for getting users, the * is the preferred wildcard, as % is not understood by LDAP (this limits the result set to records in the local database).>* `--local` *<if true, returns local users only>*	
Available fields	**Field output names**		
(see `getUser` for field descriptions)	email fullUserName	mutable providerName	userName
Usage example	`cmtool getUsers --pattern ec*` Retrieves information on all user IDs that begin with 'ec'.	

getUserSettings	
Description	Retrieves settings for the currently logged-in user.
Required arguments	None
Optional arguments	None
Usage example	`cmtool getUserSettings`

login		
Description	Logs in to the client with the appropriate credentials and creates a session file in the users home directory, which allows subsequent calls to cmtool to connect to the cluster manager.	
Required arguments	*<userName>* - The unique name of the user. *<password>* - The secret value used to identify an account for a particular user.	
Optional arguments	`--passwordFile` *<path to password file, if* `--password` *is also specified,* `--passwordFile` *overrides its value in the command line>*	
Available fields	**Field output name**	**Description**
	`sessionId`	This is a session "cookie."
Usage example	`cmtool login ec123 bobs` Logs in a user named "ec123" whose password is "bobs". **Note:** If you do not wish to expose passwords on the command line, you can omit the password from the example above. Press the Enter key after typing the command string (without the password) and you will be prompted for the password.	

logout	
Description	Logs out of the client session.
Required arguments	None
Optional arguments	None

modifyGroup	
Description	Modifies a local group.
Required arguments	*<groupName>* - The unique name of the group.
Optional arguments	`--newName` *<new group name>*
Usage example	`cmtool modifyGroup DevGroupA --newName GroupDevA`

modifyUser	
Description	Modifies a local user.
Required arguments	*<userName>* - The unique name of the user.
Optional arguments	`--fullUserName` *<the real world name of the user>* `--email` *<the associated user email address>* `--password` *<secret value used to identify an account for a particular user>* `--passwordFile` *<path to password file, if* `--password` *is also specified,* `--passwordFile` *overrides its value in the command line>*
Usage example	`cmtool modifyUser ec123 --fullUserName "Mary Smith"`

removeGroupMember	
Description	Deletes a user name from a specific group member list.
Required arguments	*<groupName>* - The unique name of the group. *<userName>* - The unique name of the user.
Optional arguments	None
Usage example	`cmtool removeGroupMember DevGroupA ec123`

setBuildEndNotification	
Description	Enables/disables notification when builds of this class end for the currently logged-in user.
Required arguments	*<buildClassId>* - A unique number assigned by Cluster Manager for each build class. Use `getBuildClasses` to retrieve a list of build class IDs. *<enabled>* - Set this to true to enable notification and to false to disable it.
Optional arguments	None
Usage example	`cmtool setBuildEndNotification 1 true` Enables build 'end notification' for build class 1.

setPermissions		
Description	Creates or modifies permissions for a user or group. The permissions are a space-separated list of permission names.	
Required arguments	*<principalType>* - Value = user or group. *<entityName>* - A user or group name in an access entry. *<permissions>* - The list of permission flags for a particular entity.	
Optional arguments	None	
Available permission flags	**Permission names**	

AgentsDelete	EMakeImpersonate	ReportsDelete
AgentsRead	EMakeInvoke	ReportsRead
AgentsWrite	MaintenanceDelete	ReportsWrite
BuildsDelete	MaintenanceRead	ResourcesDelete
BuildsRead	MaintenanceWrite	ResourcesRead
BuildsWrite	MessageLogDelete	ResourcesWrite
ClassesDelete	MessageLogRead	ServerAccess
ClassesRead	MessageLogWrite	UserModify
ClassesWrite		

Usage example	`cmtool setPermissions user ec123 "BuildsRead AgentsRead"` Restricts user ec123 to read-only privileges for builds and agents.

setUserSettings	
Description	Updates settings for the currently logged-in user.
Required	*<watchMessages>* - Indicates whether you want to receive notifications when messages

setUserSettings	
arguments	of the specified notification level arrive. Values are Y, N, y, n, Yes, No, yes, or no.
Optional arguments	`--notificationLevel` *<value can be either Info, Warning, or Error>*
Usage example	`cmtool setUserSettings true --notificationLevel Info` Sets the current user to receive notifications for 'Info' level messages.