Command Central and Platform Manager Command Reference

Version 9.6

April 2014

Document ID: CC-PM-CR-96-20140515

Table of Contents

About this Guide

This reference describes the commands in the webMethods Command Central and webMethods Platform Manager command line interface. It provides command syntax, usage notes, and examples for each command. The information in this reference is for administrators, developers, and users that want to access the Command Central and Platform Manager functionality using commands.

Command Line Syntax Conventions

Convention	Description
`{ }` (braces)	Indicates when you have choices from which you can select one item. The choices are separated with a | (pipe). For example, the following syntax indicates you can use either `--username` or `-u` for the username option: `{--username | -u}`
`[ ]` (brackets)	Indicates optional information that you are not required to supply. For example, the following indicates that the username option is not required: `[{--username | -u}]`
italic	Indicates variables for which you must supply values. For example, the following indicates that you must supply an alias name: *node_alias*
`plain text`	Indicates arguments, options, or keywords that you must supply exactly as shown, using the same combination of upper- and lower-case letters. For example, consider the following syntax: `cc exec landscape nodes` *node_alias* `generateNodeId` This syntax indicates: ■ You must supply `cc exec landscape nodes` exactly as shown.

Convention	Description
	■ You must supply `generateNodeId` exactly as shown, using the same combination of upper- and lower-case letters.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Documentation Installation

You can download the product documentation using the Software AG Installer. The documentation is downloaded to a central directory named _documentation in the main installation directory (SoftwareAG by default).

Online Information

You can find additional information about Software AG products at the locations listed below.

If you want to...	Go to...
Access the latest version of product documentation.	**Software AG Documentation website** http://documentation.softwareag.com
Find information about product releases and tools that you can use to resolve problems. See the Knowledge Center to: ■ Read technical articles and papers. ■ Download fixes and service packs (9.0 SP1 and earlier). ■ Learn about critical alerts. See the Products area to: ■ Download products. ■ Download certified samples. ■ Get information about product availability. ■ Access older versions of product documentation. ■ Submit feature/enhancement requests.	**Empower Product Support website** https://empower.softwareag.com
■ Access additional articles, demos, and tutorials.	**Software AG Developer Community for webMethods**

If you want to...	Go to...
■ Obtain technical information, useful resources, and online discussion forums, moderated by Software AG professionals, to help you do more with Software AG technology. ■ Use the online discussion forums to exchange best practices and chat with other experts. ■ Expand your knowledge about product documentation, code samples, articles, online seminars, and tutorials. ■ Link to external websites that discuss open standards and many web technology topics. ■ See how other customers are streamlining their operations with technology from Software AG.	http://communities.softwareag.com/

1 Working with Commands

Important: The command-line interface in Command Central and Platform Manager version 9.6 and above might not be fully compatible with earlier versions. To use version 9.6 and above, you might need to make changes to the scripts that you developed with earlier versions.

Performing Post Installation Configuration

The following are tasks you can perform after installation.

- Optional. Define environment variables so that you can invoke Command Central and Platform Manager commands from any location on the machine. To do so:

  1. Set the CC_CLI_HOME environment variable to the following directory:

     Software AG_directory\CommandCentral\client

     Examples

     - Windows: `set CC_CLI_HOME=C:\SoftwareAG\CommandCentral\client`
     - UNIX: `export CC_CLI_HOME="/opt/SoftwareAG/CommandCentral/client"`

  2. Add "$CC_CLI_HOME/bin" to the PATH environment variable.

     Examples

     - Windows: `set PATH=%PATH%;%CC_CLI_HOME%\bin`
     - UNIX: `export PATH="$PATH:$CC_CLI_HOME/bin"`

- Optional. Set CC_SERVER environment variable to either a Command Central or Platform Manager server endpoint address. By doing so, if you omit the `{--server | -s}` option from a command, the command uses the value you specify for the CC_SERVER variable. For more information, see "--server | -s" on page 50.

  Examples

  - Windows: `set CC_SERVER=http://rubicon:8090/cce`
  - UNIX: `export CC_SERVER="http://rubicon:8090/cce"`

- Optional. Set CC_USERNAME environment variable to a user name. By doing so, if you omit the `{--username | -u}` option from a command, the command uses the value you specify for the CC_USERNAME variable. For more information, see "--server | -s" on page 50.

  Examples

  - Windows: `set CC_USERNAME=Administrator`
  - UNIX: `export CC_USERNAME="Administrator"`

Executing Command Central Commands

To execute a Command Central command:

1. From the command prompt, change directory to the following location where the executable files for the Command Central commands reside:

   Software AG_directory\CommandCentral\client\bin

   Note: This step is not necessary if you have set the CC_CLI_HOME environment variable and included it on the PATH environment variable. For more information, see "Performing Post Installation Configuration" on page 14.

2. Enter the command you want to execute.

   For example, to list products that Command Central manages, enter:

   ```
   cc list inventory products
   ```

Executing Platform Manager Commands

There are no separate executable files for Platform Manager commands. You use the executable files for the Command Central commands, and then point to the appropriate Platform Manager server using the `{--server | -s}` option.

To execute a Platform Manager command:

1. From the command prompt, change directory to the following location where the executable files for the Command Central commands reside:

   Software AG_directory\CommandCentral\client\bin

   Note: This step is not necessary if you have set the CC_CLI_HOME environment variable and included it on the PATH environment variable. For more information, see "Performing Post Installation Configuration" on page 14.

2. Enter the command you want to execute, using the `{--server | -s}` option to identify the Platform Manager server. For more information, see "--server | -s" on page 50.

   For example, if you want to list the products that the Platform Manager server with host name rubicon2 and port number 8092 manages, enter:

   ```
   cc list inventory products --server http://rubicon2:8092/spm
   ```

   Note: If you have set the CC_SERVER environment variable to the appropriate Platform Manager server, you can omit the `{--server | -s}` option.

Return Codes from Command Execution

The following table lists the return codes that can result from executing a Command Central or Platform Manager command.

Return Code	Description
0	Indicates the execution of the command was successful. A command returns 0 when the HTTP response code is less than 400.
1	Indicates that the command syntax is not valid.
10	Indicates the output that a command returned does not match the expected values specified with the `{--expected-values |-e}` option.
response-code	Indicates the command execution resulted in an error. The return code is the HTTP response code. A command uses this return code when the HTTP response code is greater than or equal to 400. **Note:** On Unix, return codes greater than 256 are considered "Out of range exit values." As a result, when executing the commands on a client machine that runs on Unix, if the HTTP response code is greater than 256, the return code is *response-code* modulo 256.

Summary of Commands

The following tables lists the commands available in the command line interface. The table also indicates whether a command is only applicable for Command Central, only applicable for Platform Manager, or can be executed on both Command Central and Platform Manager servers.

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc get configuration common	Retrieves the schema for a specified configuration type.		X

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc get configuration compare	Compares a configuration type used by two or more run-time components.	X	
cc create configuration data	Creates a new instance of a configuration type.	X	X
cc delete configuration data	Deletes a configuration instance.	X	X
cc get configuration data	Retrieves data for a configuration instance.	X	X
cc update configuration data	Updates the data for a configuration instance.	X	X
cc get configuration instances	Retrieves information about a configuration instance.	X	X
cc list configuration instances	Lists the configuration instances.	X	X
cc get configuration types	Retrieves information for one or more configuration types.	X	X
cc list configuration types	Lists information for one or more configuration types.	X	X
cc exec configuration validation create	Validates the configuration instance data in a supplied input file.	X	X
cc exec configuration validation create	Determines whether a configuration instance can be deleted.	X	X

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc exec configuration validation create	Validates the configuration instance data in the supplied input file to determine whether you can use it to update a specified configuration instance.	X	X
cc get diagnostics logs	Retrieves log entries from a log file.	X	X
cc get diagnostic logs export file	Exports one or more log files for a specified run-time component in a zip archive file.	X	
cc list diagnostics logs	Lists the log files that a specified run-time component supports.	X	X
cc get inventory components	Retrieves information about a run-time component.	X	X
cc list inventory components	Lists information about run-time components.	X	X
cc update inventory components	Updates the display name and/or icon associated with a run-time component.	X	
cc get inventory fixes compare	Compares the fixes installed in two or more installations.	X	
cc list inventory fixes	Lists information about fixes that have been applied to products.	X	X

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc get inventory products compare	Compares the products installed in two or more installations.	X	
cc get inventory products	Retrieves information about a product.	X	X
cc list inventory products	Lists information about products.	X	X
cc create instances	Creates a new instance of an installed product.	X	X
cc delete instances	Deletes an existing instance of an installed product.	X	X
cc list instances supported products	Retrieves a list of products that support instance management.	X	X
cc update instances	Updates configuration properties of an existing instance of an installed product.	X	X
cc list jobmanager jobs	Lists information about long-running jobs.		X
cc add landscape environments nodes	Adds one or more existing installations to an environment.	X	
cc create landscape environments	Creates a new environment.	X	
cc delete landscape environments	Deletes an environment.	X	

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc get landscape environments	Retrieves information about an environment.	X	
cc list landscape environments	Lists environments in the landscape.	X	
cc remove landscape environments nodes	Removes one or more installations from an environment.	X	
cc update landscape environments	Updates the display name and/or description assigned to an environment.	X	
cc create landscape nodes	Adds an installation that you want to manage via Command Central.	X	
cc delete landscape nodes	Removes an installation from being centrally managed via Command Central.	X	
cc exec landscape nodes generateNodeId	Generates or regenerates a unique ID for an existing installation.	X	
cc get landscape nodes	Retrieves information about an installation.	X	
cc list landscape nodes	Lists the installations that Command Central manages.	X	
cc update landscape nodes	Updates the properties assigned to an installation, for example, the display name or description.	X	

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc create license-tools reports snapshot	Creates a license compliance snapshot report based on the currently registered nodes in a Command Central instance.	X	
cc delete license-tools reports snapshot	Deletes all generated license reports from Command Central.	X	
cc delete license-tools reports snapshot reportid	Deletes an existing license report with the specified unique report identifier.	X	
cc get license-tools reports snapshot	Lists all license reports available on the Command Central server.	X	
cc get license-tools reports snapshot reportid	Obtains information about a license report with the specified unique report identifier.	X	
cc get license-tools reports snapshot output PDF	Generates a PDF file for an existing license report.	X	
cc get license-tools reports snapshot output XML	Generates an existing license report in XML format.	X	
cc exec lifecycle	Executes an action to start, stop, pause, and/or resume run-time components.	X	X
cc get monitoring	Retrieves the run-time statuses, run-time states, or states of run-time components.	X	X

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc list monitoring alerts	Lists the alerts for a specified run-time component.	X	
cc add repository fixes	Registers a fix repository.	X	X
cc add repository products	Registers product repositories.	X	X
cc create repository	Creates a product or fix repository.	X	
cc delete repository	Deletes a registered product or fix repository.	X	
cc delete repository with node_alias	Deletes a registered product or fix repository on a specified Platform Manager node. The command removes access to the repository without deleting actual repository data.	X	X
cc delete repositories	Deletes all registered product or fix repositories.	X	
cc exec repository discover	Finds product and fix repositories for the specified host, name, and port and adds the discovered repositories to Command Central.	X	X
cc exec repository discover with node_alias	Discovers public product or fix repositories.	X	X
cc exec repository products register	Registers a product repository on a specified	X	

Command	Command Description	Supported on Command Central	Supported on Platform Manager
	Platform Manager node and deletes all previously added repositories on the node.		
cc exec repository register	Copies a product or fix repository, including its image file, to a new Platform Manager node.	X	
cc list repository	Lists registered product or fix repositories.	X	
cc list repository with node_alias	Lists registered product and fix repositories on a specified Platform Manager node.	X	X
cc list repository discover	Finds product or fix repositories for the specified host, name, and port, but does not add the discovered repositories to Command Central.	X	
cc update repository	Updates a repository using data from an XML file.	X	
cc update repository details	Updates a repository description and location.	X	
cc list resources icons	Lists information about the installed icons.	X	
cc add security credentials	Adds security credentials.	X	
cc delete security credentials	Deletes security credentials.	X	

Command	Command Description	Supported on Command Central	Supported on Platform Manager
cc get security credentials	Retrieves security credentials.	X	
cc create templates	Creates a new template for an existing managed installation.	X	X
cc delete templates	Removes a template from an installation.	X	
cc exec templates apply	Applies a template registered and available in a managed installation.	X	X
cc export templates	Exports an existing template.	X	X
cc list templates	Lists all templates available in a landscape.	X	X
cc exec templates import	Registers an exported template.	X	X

Getting Familiar with Using Commands

The following illustrates how you might get familiar using the Command Central and Platform Manager commands. For more information about the commands used in the examples in the following table, see"Landscape Commands" on page 129.

Step	Description	Examples
1.	Use a `list` command to view the type of information the command returns.	Execute the following command to view a list of installations. `cc list landscape nodes` The output includes alias names for all the installations. You can use the alias names in subsequent commands to get data for an installation, update an

Step	Description	Examples
		installation, execute actions against an installation, or delete an installation.
2.	Use a `get` or `list` command to retrieve information for a specific instance. **Note:** The `get` and `list` commands are equivalent.	In this example, assume the `list` command provided information for an installation that has the alias name "sag01". To retrieve information for the "sag01" installation, returning the information to an output file in XML format, execute the following command: `cc get landscape nodes sag01 --output info --format xml`
3.	Use a `create` command to create a new instance. You can edit the output file that a `get` command returns to specify the information for the new instance. Then you can use that file as input to the `create` command.	To create a new installation with alias name "new", edit the info.xml file that the `get` command returned to supply the alias name, URL, and description for the new installation. Then execute the following command: `cc create landscape nodes --input info.xml` **Note:** If you execute the `list` command again, the command lists the "new" installation.

4.	Use an `update` command to update data for an instance.
	
	You can use a `get` command to retrieve information for the specific instance you want to update, returning the output to a file. Then you can update the output file the `get` command returns and use that as the input to the `update` command.
		For this example, update the "new" installation. Execute the following command to retrieve information for the "new" installation, returning the output to a file in XML format:
		
		`cc get landscape nodes new --output updatefile --format xml`
		
		Update the data in the returned "updatefile". For example, you might specify a new description. Then execute the following command to update the installation information:
		
		`cc update landscape nodes new --input updatefile`

5.	Use an `exec` command to execute an action against an instance.	To generate a new ID for the "new" installation, execute the following command: `cc exec landscape nodes new`

Step	Description	Examples
		`generateNodeId`
6.	Use a `delete` command to remove an instance.	To delete the "new" installation, execute the following command: `cc exec landscape nodes new`

Note: Based on the resource you are working with, all types of commands, that is `list`, `get`, `create`, `update`, `exec`, and `delete`, might not be available.

Displaying Help for the Command Line Interface

You can display help for the command line interface tool from the command prompt.

To display help for the command line interface tool.

- To display general help that includes operations and common options, enter `cc` with no other arguments. For example:

  ```
  cc
  ```

- To display a list of Command Central commands, including the syntax of the commands, use the `{--help | -h}` option. Also include the `{--server | -s}` option to identify a Command Central server. For example:

  ```
  cc --help --server http://rubicon:8090/cce
  ```

  Note: If you omit the `{-server | -s}` option, the command uses the value from the `CC_SERVER` environment variable.

- To display a list of Platform Manager commands, including the syntax of the commands, use the `{--help | -h}` option. Also include the `{--server | -s}` option to identify a Platform Manager server. For example:

  ```
  cc --help --server http://spm:8092/spm
  ```

  Note: If you omit the `{-server | -s}` option, the command uses the value from the `CC_SERVER` environment variable.

2 Securing Communication with the Command Central Server

Configuring SSL Using Command Options

You can secure connections to the Command Central server by using Command Line interface options that contain SSL configuration settings, such as the location of the keystore and truststore files. For more information about the SSL-related command options, see "Options for the Commands" on page 31.

Configuring SSL Using Configuration Properties Files

Command Central comes with a default configuration properties file that contains the SSL command options with the default SSL settings. When you have set the CC_CLI_HOME and the PATH environment variables, the default configuration settings in the CC_CLI_HOME\conf\cc.properties file are used by the local Command Central server to communicate over HTTPS.

You should make a copy of the default configuration properties file to create a custom file in which you set all SSL-related configuration settings. Using the `--configuration-file` command option, you specify the location of the custom configuration properties file. For more information about the command option, see "--configuration-file" on page 34.

Creating a Custom Command Central Properties File

Important: Software AG does not recommend editing or changing the settings in the default cc.properties file, located in the *Software AG_directory*\CommandCentral\client\conf directory. Use the following procedure to create a custom configuration properties file and set the required authentication settings in the custom file.

To create a custom properties configuration file

1. Go to *Software AG_directory*\CommandCentral\client\conf
2. Copy the cc.properties files to the following location: *user_home*\.sag\cc.properties

   Note: To create the .sag directory in Windows, at the command prompt type `mkdir %HOME%\.sag`
3. Set file permissions for your copy of the cc.properties file to prevent other users from accessing the file.
4. Edit the custom cc.properties file in a text editor as required.

   The following table lists the default configuration settings and the option that you can use to override the default value:

Property	Default Value	Use this option to override the default setting
server	`https://localhost:8091/cce`	`--server`
username	`Administrator`	`--user`
password		`--password`
ssl-truststore-file	`demo-truststore.jks`	`--ssl-truststore-file`
ssl-truststore-password		`--ssl-truststore-password`
ssl-trust-all-hosts		`--ssl-trust-all-hosts`

For more information about the command options listed in the table, see "Options for the Commands" on page 31.

5. Save the custom cc.properties file.

Configuring SSL Using Scripts

You can secure connections to the Command Central server by using ANT properties. For more information about the SSL-related ANT properties, see "Parameters to Use with the ccsetup Task" on page 269.

Considerations When Using Configuration Properties

Determining the Value for Configuration Properties

The following lists the order used to determine the value for any of the configuration properties when executing a CLI command:

1. Value in the first command option or ANT property.
2. Value in the custom cc.properties file in the *user_home* \.sag\cc.properties directory.
3. Value in the default cc.properties file in the CC_CLI_HOME\conf directory if you have set the CC_CLI_HOME and the PATH environment variables.

Specifying the Password

When executing a command using the Command Line interface, Command Central prompts for a password each time the `cc` command is executed. You must specify a password for your user and truststore in one of the following:

- The `--password` and `--ssl-truststore-password` options
- The default or custom cc.properties configuration files
- The CC_PASSWORD environment variable

3 Options for the Commands

Common Options

The following are options that the Command Line Interface (CLI) supports. To determine the options that a specific command allows, see the documentation for that command.

Note: When you use both a deprecated option and the new option that replaces it in the same command, the new option overrides the value of the deprecated option.

--accept | -a

Deprecated. Use `--output-format | -f` in place of `--accept | -a`. When you use `--accept | -a`, Command Central executes the command with a warning.

Specifies the format you want the command to use for the data it returns. Use the `{--accept | -a}` option to specify a content type that the command supplies on the HTTP Accept request header that it sends to Command Central or Platform Manager.

Syntax

`{--accept | -a}` *content_type*

Arguments and Options

Argument	Description
content_type	Specifies a well-formed content type that indicates the format you want the command to use for the output. The following lists some examples: ■ application/xml ■ application/json ■ text/plain ■ text/tab-separated-values ■ text/csv

Usage Notes

- If you specify the `{--input | -i}` option, the command ignores the `{--accept | -a}` option and sets the request content type based on the file extension of the input file. For more information, see "--input | -i" on page 42.
- Use the `{--accept | -a}` option as an alternative to the `{--format | -f}` option. Both options set the request content type.
- If you specify both the `{--accept | -a}` option and the `{--format | -f}` option, the command uses the content type you specify with the `{--accept | -a}` option and ignores the `{--format | -f}` option.
- By default, output is written to the console. If you want the output written to a file, use the `{--output | -o}` option. For more information, see "--output | -o" on page 47.

Examples

- To have a command return data in JavaScript Object Notation format:

  ```
  --accept application/json
  ```

- To have a command return data in csv format:

  ```
  --accept text/csv
  ```

--check-every | -c

Specifies how often (in seconds) to check whether a long-running operation has returned the expected values. Use in conjunction with the --expected-values | -e and --wait | -w options.

Syntax

```
{--check-every | -c} seconds
```

Arguments

Argument	Description
seconds	Specifies the number of seconds the command waits before checking for expected output specified by the `{--expected-values | -e}` option. If you omit the `{--check-every | -c}` option, the command uses the value of the `CC_CHECK_EVERY` environment variable. If the `CC_CHECK_EVERY` environment variable is not set, the command uses 15 seconds.

Usage Notes

- The `{--check-every | -c}` option is only needed when you specify the `{--expected-values | -e}` option.
- The command is continually executed every `{--check-every | -c}` seconds until the command either returns the expected values or times out because the seconds specified by the `{--wait | -w}` option have elapsed.
- If the time specified by the `{--wait | -w}` option elapses before the expected results are returned, the command fails.
- The use of the `{--expected-values | -e}`, `{--wait | -w}`, and `{--check-every | -c}` options is helpful with commands that perform actions that might take several seconds or minutes to complete. Depending on your use case, these options might be helpful with any command. However, they are most helpful with the `lifecycle` and `monitoring` commands because they allow you to reliably execute the commands.

Example

To have a command check every 30 seconds for the expected results:

```
--check-every 30
```

Note: To see an example that uses all of the `{--expected-values | -e}`, `{--wait | -w}`, and `{--check-every | -c}` options, see "--expected-values | -e" on page 37.

--configuration-file

Specifies the location of the configuration file that contains a list of configuration properties, such as SSL, server, username, and password settings.

Syntax

```
--configuration-file path
```

Arguments

Argument	Description
path	Specifies the fully qualified path to the location of the configuration properties file. The default Command Central cc.properties file is located in the *Software AG_directory*\CommandCentral\client\conf directory. **Note:** You can specify a relative path to the current execution directory of the cc command if you have set the CC_CLI_HOME and PATH environment variables.

Usage Notes

- When you include both the `--configuration-file` and the `-ssl-truststore-password` options, Command Central uses the password specified in the `-ssl-truststore-password` option.
- If you do not want to specify the SSL truststore options for each command execution, you can include the`--ssl-truststore-file`, `--ssl-truststore-password`, and `--ssl-trust-all-hosts` options in a custom cc.properties file and specify the path to the location of that file in the `--configuration-file` option. For more information about creating a custom configuration properties file, see "Configuring SSL Using Configuration Properties Files" on page 28.

--debug | -d

Specifies you want the command to return extra information that you can use for debugging issues, in addition to the returning service output. The extra information includes:

- HTTP service request
- URL of the Command Central or Platform Manager server to which the request was sent
- Request content type
- Accept header for the request
- HTTP response code from the request
- Response content type
- Response content length

Syntax

```
{--debug | -d}
```

Arguments

None

Usage Notes

- If you specify both `{--debug | -d}` and `{--quiet | -q}`, the command ignores the `{--quiet | -q}` option and uses the `{--debug | -d}` option to display the additional debug information.

Example

The following shows sample output that uses the `--debug` option.

```
cc list landscape nodes --debug
```

--error | r

Specifies a file where you want a command to write the output if the command results in an error. If you do not specify the `{--error | -r}` option, the command writes the output to the console.

Syntax

```
{--error | -r} file
```

Arguments

Argument	Description
file	Specifies the file where you want the error output written. If the file you specify does not exist, the command creates it. You can specify: - Absolute directory path and filename. - Relative directory path and filename. The path is relative from where you initiated the command.

Argument	Description
	■ Filename of a file in the same directory where you initiated the command.

Usage Notes

- If the file you specify with the `{--error | -r}` option already exists, the command overwrites the existing file with the new service results.
- If a command encounters an error, to help resolve errors, you can execute the command again using the `{--debug | -d}` option to display additional information about the actual request and response.
- You can use the `{--error | r}` option to direct error results to a specific location, for example, if you want to use automated tools to review output.
- If a command executes successfully, the command writes the output to the location specified by the `{--output | -o}` option or the console if the `{--output | -o}` option is not specified.

Examples

- To write error output to a file named “errors.xml” in the directory c:\outputs:

  ```
  --error c:\outputs\errors.xml
  ```

- To write error output to a file named “errors.json” in the \outputs directory relative to where you initiate the command:

  ```
  --error outputs\errors.json
  ```

- To write output to a file named “errors” in the same directory from where you initiate the command:

  ```
  --error errors
  ```

  In this example, the command determines the file extension based on the request content type.

--expected-values | -e

Specifies the expected values for which to wait before a command completes. Use in conjunction with the --expected-values | -e and --wait | -w options.

When you use the `{--expected | -e}`, `{--check-every | -c}`, and `{--wait | -w}` options, each `{--check-every | -c}` seconds the command is executed and the results are examined for the values specified with the `{--expected | -e}` option. The command is successful if the command returns the expected values within the wait time. The command fails if the command does not return the expected values within the wait time.

Syntax

```
{--expected-values | -e} values
```

Arguments

Argument	Description

values	Specifies the values that must be present in the output for a command to complete. Use a comma to separate each value. If you use a value that includes spaces, place quotes around the value, for example:
	`--expected-values "a value with spaces"`
	If you use a value that includes a logical OR operator, use | as a separator, for example the following command checks if the output contains either DONE or WARNING:
	`--expected "DONE|WARNING"`

Usage Notes

- The use of the `{--expected-values | -e}`, `{--wait | -w}`, and `{--check-every | -c}` options is helpful with commands that perform actions that might take several seconds or minutes to complete. Depending on your use case, these options might be helpful with any command. However, they are most helpful with the `lifecycle` and `monitoring` commands because they allow you to reliably execute the commands.
- If you specify multiple values with the `{--expected-values | -e}` option, the command checks for *all* values and returns successfully only if *all* of them are present. For example, if you specify `STOPPED,UNKNOWN,ONLINE`, the command first checks for the `STOPPED` run-time status. Once stopped, the command checks for the `UNKNOWN`, and then after that it checks for `ONLINE`. If any of the run-time statuses do not occur before the command times out, the command returns an error indicates the missing statuses.
- If you specify the `{--expected-values | -e}` option, but omit the `{--check-every | -c}` option, the command uses the value from the `CC_CHECK_EVERY` environment variable. If the `CC_CHECK_EVERY` environment variable is not set, the command uses 15 seconds.
- If you specify the `{--expected-values | -e}` option, but omit the `{--wait | -w}` option, the command uses the value from the `CC_WAIT` environment variable. If the `CC_WAIT` environment variable is not set, the command uses 15 seconds.

Example

To wait 180 seconds for a command to return the value “STOPPED”, then “ONLINE”, checking every 30 seconds for the expected results:

```
--expected-values STOPPED,ONLINE --wait 180 --check-every 30
```

--force

Forces the execution of a delete command without prompting for confirmation of the requested operation.

Syntax

```
--force
```

Arguments

None.

Usage Note

When you omit the `--force` option, the delete command prompts you to confirm the requested operation.

--format | -f

Deprecated. Use `--output-format | -f` in place of `--format | -f`. When you use `--format | -f`, Command Central executes the command with a warning.

Specifies the format you want a command to use for the data it returns. Command Central and Platform Manager support the following formats:

- Tab-separated values (tsv)
- Plain text (txt)
- XML (xml)
- Comma-separated values (csv)
- JavaScript Object Notation (json)

Although Command Central and Platform Manager support these formats, a specific command might only support a subset of the formats. Refer to the documentation for a specific command to determine the exact formats that it supports.

Syntax

```
{--format | -f} {tsv args | text | xml | csv args | json}
```

Arguments

Argument	Description
`tsv args`	Specifies you want the output in tab-separated values format. For more information about the arguments you can specify,

Argument	Description
	see "Arguments for Tab-Separated Values (tsv) and Comma-Separated Values (csv)" on page 41.
`text`	Specifies you want the output in plain text format.
`xml`	Specifies you want the output in XML format.
`csv` *args*	Specifies you want the output in comma-separated values format. For more information about the arguments you can specify, see "Arguments for Tab-Separated Values (tsv) and Comma-Separated Values (csv)" on page 41.
`json`	Specifies you want the output in JavaScript Object Notation format.

Usage Notes

- Use the `{--format | -f}` option as an alternative to the `{--accept | -a}` option. Both options set the request content type. For more information, see "--accept | -a" on page 32.
- If you specify both the `{--format | -f}` option and the `{--accept | -a}` option, the command uses the content type you specify with the `{--accept | -a}` option and ignores the `{--format | -f}` option.
- If you specify the `{--input | -i}` option, the command ignores the `{--format | -f}` option and sets the request content type based on the file extension of the input file. For more information, see "--input | -i" on page 42.
- By default, output is written to the console. If you want the output written to a file, use the `{--output | -o}` option. For more information, see "--output | -o" on page 47.
- The the following describes the typical default that a command uses if you do not specify the `{--format | -f}` option:
  - If you execute the command from the command line, a batch script, or a shell script, the default format is tab-separated values (tsv) format.
  - If you execute the command from an Ant script, the default format is XML format.

  To determine the default for a command that does not support tab-separated values (tsv) or XML format, refer to the documentation for that command.
- If a command supports either the `tsv` or `csv` format, you can restrict the fields the command returns to only those fields you specify. For more information, see

"Arguments for Tab-Separated Values (tsv) and Comma-Separated Values (csv)" on page 41.

Examples

- To have a command return data in csv format without headers:

  ```
  --format csv includeHeaders=false
  ```

- To have a command return data in xml format:

  ```
  --format xml
  ```

Arguments for Tab-Separated Values (tsv) and Comma-Separated Values (csv)

When you specify the `tsv` or `csv` with the `{--format | -f}` option, you can specify additional arguments to customize the output.

Syntax

```
{--format | -f} {tsv | csv} [includeHeaders={labels | properties | none}]
[properties=keys ]
```

Argument Descriptions

Argument	Description
`[includeHeaders={labels | properties | none}]`	Specifies whether you want the output to include a header line. Specify one of the following: - `labels` to include a header line containing the display names for each field. For example "Product Version" might be a display name if the output includes the version of a product. This is the default. - `properties` to include a header line containing the property key name for each field. For example "product.version" might be the display name if the output includes the version of a product. - `none` to omit headers from the output.
`[properties=keys]`	Identifies the keys associated with the information you want included in the output. For example, if you want the output to only include the product version, specify `properties=product.version`.

Argument	Description
	To specify multiple keys, separate each with a comma. For example, if you want alias names and descriptions in the output, you might specify `properties=alias,description`. Use `properties=*` to include all information. If you omit the `properties` argument, the command returns a default set of fields.

Usage Notes

- To determine the keys you can specify with the `properties` argument, execute a `get` or `list` command and specify `includeHeaders=properties properties=*` so that the output displays a header line that shows the keys for all the possible properties.

  For example, you might want to use the `cc list landscape nodes` command to retrieve the list of alias names and descriptions for installations. First, execute the `cc list landscape nodes` command with `--format csv includeHeaders=properties properties=*` to determine that the key for the alias name is `alias` and the key for the description is `description`. You can then execute `cc list landscape nodes` with `--format csv includeHeaders=name properties=alias,description`.

--input | -i

Identifies a file that contains the input data for a `create`, `add`, `update`, or `exec` command.

For example, when using the `cc create landscape nodes` command to add a new installation that you want Command Central to manage, you are required to provide an alias name for the installation and the URL for the installation. You can provide this information on the command line using command line arguments, or you can use the `{--input | -i}` option to specify this data in an input file. For some commands, the item you are creating, adding, or updating requires more data than is practical to supply on the command line, and as a result, the `{--input | -i}` option might be required to supply the data for the command.

Syntax

`{--input | -i}` *`filename`*`{.xml | .json | .properties}`

Arguments

Argument	Description
`filename``{.xml	.json	.properties}`	Specifies the file that contains the input data. The input file can be:

Argument	Description
	■ An .xml file containing input data in XML format ■ A .json file containing input data in JavaScript Object Notation format ■ A .properties file containing input data in key/value pairs format. When identifying the input file, you can specify: ■ Absolute directory path and filename. ■ Relative directory path and filename. The path is relative from where you initiated the command. ■ Filename of a file in the same directory where you initiated the command.

Usage Notes

- The use of an input file for data is helpful when:
  - You are scripting commands, for example, using an Ant script.
  - You are executing a command with complicated input parameters where it is easier to specify them in a file in XML or json format rather than specifying them on the command line.
  - You want to create templates for adding items such as installations, configurations, etc.
- A command always sets the request content type based on the file extension of the input file if the `{--input | -i}` option is specified. This is true even if you specify another option that affects the request content type, that is, the `{--accept | -a}` or `{--format | -f}` option.

  The following lists the request content type that a command uses based on the file extension of the input file:

File extension	Request content type
.xml	application/xml
.json	application/json
.properties	application/x-www-form-urlencoded

- The `{--input | -i}` option is supported for POST and PUT requests, that is for `create`, `add`, `exec` and `update` commands. It is not supported for GET and DELETE requests, that is `get`, `list`, `delete`, or `remove` commands.
- The input file contains data that a command requires for creating an item, for updating an item, or for the execution of an operation. You must supply a file in the format that the command expects, using specific element names and/or tags. For example, when using the `cc create configuration data` command to create an instance of a COMMON-PORTS configuration type, to supply the port number in an XML file, you might include the element `<Number>5555</Number>` as part of the XML file.

  To determine the format to use for an input file, execute a `get` or `list` command to retrieve a similar item. On the `get` or `list` command, if you use the `{--output | -o}` option to write the output to a file, you can then update the returned output file and specify it with the `{--input | -i}` option as an input data file.

  For example, if you want to use the `cc create configuration data` command to create a COMMON-PORTS configuration instance, first use the `cc get configuration data` command to retrieve an existing COMMON-PORTS instance to learn the format to use for the input data file.
- If you specify input data both on the command line and use the `{--input | -i}` option to specify data in an input file, the command uses the data that you specify in the input file and ignores the data you specify on the command line.

Examples

- To use the input file input.xml in the directory c:\templates:

  ```
  --input c:\templates\input.xml
  ```
- To use the input file input.xml in the \templates directory relative to where you initiate the command:

  ```
  --input templates\input.xml
  ```
- To use the input file input.xml that resides in the same directory from where you initiate the command:

  ```
  --input input.xml
  ```

--input-format | -m

Specifies the content type of the input data for a command. You can specify the same values for `--input-format | -m` and `--output-format | -f`.

Syntax

```
--input-format | -m content-type
```

Arguments

Argument	Description
content-type	Specifies a well-formed content type that indicates the format you want the command to use for the input. You can specify the short or full versions of the media type. The following lists some examples: ■ application/xml | xml ■ application/json | json ■ text/plain | text ■ text/tab-separated-values | tsv ■ text/csv | csv The default value is taken from the input file extension if the extension matches the short version of a supported media type. If the input file extension does not match the short version of a supported media type, the default is text/plain. For the content types that a command supports, see the documentation for a specific command.

Example

To specify the input data is content type application/xml:

```
--input-format application/xml
```

--log | -l

Specifies a file where you want to log the outcome from the execution of the command, whether the command completes successfully or encounters errors. If you do not specify the `{--log | -l}` option, the command logs this error information to the console.

The logged results include:

- Service output
- Errors that occur while interpreting the command

  Note: If the error occurs while the initializing the command, the error is written to the console rather than the file specified with the `{--log | -l}` option

- Debug information if the `{--debug | -d}` option is specified on the command

Syntax

```
--log | -l file
```

Arguments

Argument	Description
file	Specifies the log file where you want the errors written. If the file you specify does not exist, the command creates it. You can specify: ■ Absolute directory path and filename. ■ Relative directory path and filename. The path is relative from where you initiated the command. ■ Filename of a file in the same directory where you initiated the command.

Usage Notes

- If you use the `{--output | -o}` option with the `{--log | -l}` option and the command completes successfully, the command writes the results to the output file *and* logs the outcome to the log file.
- If you use the `{--error | -r}` option with the `{--log | -l}` option and the command encounters an error, the command writes the error results to the error file *and* logs the outcome to the log file.
- If a command uses the `{--debug | -d}` command, the debug information is also written to the log file.
- If the file you specify with the `{--log | -l}` option already exists, the command appends the new service results to the file.
- The error information includes a timestamp. Using this option for commands generates a history of the command execution and actions.

Examples

- To log information to a file named “logfile.xml” in the directory c:\outputs:

  ```
  --log c:\outputs\logfile.xml
  ```

- To log information to a file named “logfile.json” in the \outputs directory relative to where you initiate the command:

  ```
  --log outputs\logfile.json
  ```

- To log information to a file named “logfile” in the same directory from where you initiate the command:

  ```
  --log logfile
  ```

--media-type | -m

Deprecated. Use `--input-format | -m` in place of `--media-type | -m`. When you use `--media-type | -m`, Command Central executes the command with a warning.

Specifies the content type of the input data.

Syntax

`--media-type | -m` *`content-type`*

Arguments

Argument	Description
`content-type`	Specifies the content type. Refer to the documentation for a specific command to determine the content types that a command supports.

Example

To specify the input data is content type application/xml:

```
--media-type application/xml
```

--output | -o

Specifies a file where you want a command to write the output if the command executes successfully. If you do not specify the `{--output | -o}` option, the command writes the output to the console.

Syntax

`--output | -o` *`file`*

Arguments

Argument	Description
`file`	Specifies the file where you want the output written. If the file you specify does not exist, the command creates it. You can specify: ■ Absolute directory path and filename. ■ Relative directory path and filename. The path is relative from where you initiated the command.

Argument	Description
	■ Filename of a file in the same directory where you initiated the command.

Usage Notes

- If the file you specify with the `{--output | -o}` option already exists, the command overwrites the existing file with the new service results.
- If a command results in an error, the command writers the error output to the location specified by the `{--error | -r}` option or the console if the `{--error | -r}` option is not specified.

Examples

- To write output to a file named "results.xml" in the directory c:\outputs:

  ```
  --output c:\outputs\results.xml
  ```

- To write output to a file named "results.json" in the \outputs directory relative to where you initiate the command:

  ```
  --output outputs\results.json
  ```

- To write output to a file named "results" in the same directory from where you initiate the command:

  ```
  --output results
  ```

  In this example, the command determines the file extension based on the request content type.

--output-format | -f

Specifies the content type of the output data for a command. You can specify the same values for `--input-format | -m` and `--output-format | -f`.

Syntax

```
--output-format | -f content-type
```

Arguments

Argument	Description
content-type	Specifies a well-formed content type that indicates the format you want the command to use for the output. You can specify the short or full versions of the media type. The following lists some examples: ■ application/xml | xml

Argument	Description
	■ application/json | json ■ text/plain | text ■ text/tab-separated-values | tsv ■ text/csv | csv The default value is taken from the output file extension if the extension matches the short version of a supported media type. If the output file extension does not match the short version of a supported media type, the default is text/tab-separated-values. For the content types that a command supports, see the documentation for a specific command.

Example

To specify the output data is content type application/xml:

```
--output-format application/xml
```

--password | -p

Specifies the password for the user executing the command.

Syntax

```
{--password | -p} password
```

Usage Notes

- If you omit the `{--password | -p}` option, the command uses the value from the `CC_PASSWORD` environment variable. If the `CC_PASSWORD` environment variable is not set, the command prompts the user for the password.

Example

To specify the password “secret”:

```
--password secret
```

--quiet | -q

Specifies that you want the command to return only service output with no additional information.

Syntax

```
{--quiet | -q}
```

Arguments

None.

Usage Notes

- If you specify both `{--debug | -d}` and `{--quiet | -q}`, the command ignores the `{--quiet | -q}` option and uses the `{--debug | -d}` option to display additional debug information.

--server | -s

Identifies the Command Central or Platform Manager server on which to execute a command.

Syntax

```
{--server | -s} url
```

Arguments

Argument	Description

url	Identifies the URL of a Command Central or Platform Manager server on which to execute the command.
	- When specifying the URL of a Command Central server, if you omit:
	  - Protocol, the command uses “https://”.
	  - Port number, the command uses “8090”, which is the default port for a Command Central server.
	  - Context, the command uses “cce”.
	- You must always provide the full URL, including the protocol, when identifying a Platform Manager server.

Usage Notes

- If you omit the `{--server | -s}` option, the command uses the value from the `CC_SERVER` environment variable. If the `CC_SERVER` environment variable is not set, the command executes on localhost:8090.

- If you want to execute a command on a Platform Manager server, either specify the `{--server | -s}` option on the command or ensure the `CC_SERVER` environment variable specifies a Platform Manager server.

Examples

- To execute a command on the Command Central server with host name rubicon and port number 8090 using the http protocol:

```
--server rubicon
--server rubicon:8090
--server http://rubicon:8090
--server http://rubicon:8090/cce
```

- To execute a command on the Platform Manager server with host name rubicon2 and port number 8092 using the http protocol:

```
--server http://rubicon2:8092/spm
```

--ssl-truststore-file

Specifies the location of the truststore file.

Syntax

```
--ssl-truststore-file=path
```

Arguments

Argument	Description
path	Specifies the fully qualified path to the truststore location. The default Command Central truststore with name demo-truststore.jks is located in the *Software AG_directory*\CommandCentral\client\conf directory. **Note:** You can specify a relative path to the current execution directory of the cc command if you have set the CC_CLI_HOME and PATH environment variables.

Usage Notes

If you do not include the `--ssl-truststore-file` option, the command fails to execute.

Example

To execute the command with the Command Central default truststore:

```
--ssl-truststore-file=Software AG_directory\CommandCentral\client\
conf\demo-truststore.jks
```

--ssl-trust-all-hosts

Specifies whether to trust all hosts. When the option is included in the command, Command Central does not verify the name of the server host.

Syntax

```
--ssl-trust-all-hosts
```

Arguments

None.

Usage Notes

- The default Platform Manager and Command Central keystore with name demo-keystore.jks contains a signed CA certificate. However, the generated CA certificate does not contain the fully qualified name of the server host required by the client to trust the server. When the `ssl-trust-all-hosts` option is included, Command Central does not verify the name of the server host.
- If you do not include the `--ssl-trust-all-hosts` option, Command Central will attempt to verify the server host name, and the client will not trust a server certificate without a fully qualified server host name. When the server host name matches the host name in the signed CA certificate, the option is ignored.

--ssl-truststore-password

Specifies the password for the truststore.

Syntax

```
--ssl-truststore-password=password
```

Usage Notes

- When you do not provide a truststore password or you specify the wrong password, the command fails.
- When you include both the `--configuration-file` and the `-ssl-truststore-password` options in the command, Command Central uses the password specified in the `-ssl-truststore-password` option.

--username | -u

Specifies the user who is executing a command. The specified user must have the proper authorization to execute the command.

Syntax

```
{--username | -u} user_name
```

Arguments

Argument	Description
user_name	Specifies the user name of the user executing the command.

Usage Notes

- If you omit the `{--username | -u}` option, the command uses the value from the `CC_USERNAME` environment variable. If the `CC_USERNAME` environment variable is not set, the command uses "Administrator".
- Use the `{--password | -p}` option to specify the user's password. If you omit the `{--password | -p}` from the command line, the command will prompt you for the password. For more information, see "--password | -p" on page 49

Example

To execute the command as the user with user name admin02:

```
--username admin02
```

--wait | -w

Specifies how many seconds to wait for a long-running operation to return the expected values. Use in conjunction with the --expected-values | -e and --check-every | -c options.

Syntax

```
{--wait | -w} seconds
```

Arguments

Argument	Description
seconds	Specifies the number of seconds the command waits for the expected output specified by the `{--expected-values	-e}` option before completing.  The default is the value of the `CC_WAIT` environment variable. If the `CC_WAIT` environment variable is not set, the command uses 120 seconds.

Usage Notes

- The `{--wait | -w}` option is only needed when you specify the `{--expected-values | -e}` option.
- If the time specified by the `{--wait | -w}` option elapses before the expected results are returned, the command fails.
- The use of the `{--expected-values | -e}`, `{--wait | -w}`, and `{--check-every | -c}` options is helpful with commands that perform actions that might take several seconds or minutes to complete. Depending on your use case, these options might be helpful with any command. However, they are most helpful with the `lifecycle` and `monitoring` commands because they allow you to reliably execute the commands.

Example

To have a command wait 180 seconds for the expected results:

```
--wait 180
```

Note: To see an example that uses all of the `{--expected-values | e}`, `{--wait | -w}`, and `{--check-every | -c}` options, see "--expected-values | -e" on page 37.

4 Configuration Commands

cc get configuration common

Retrieves the schema for a specified configuration type.

Syntax

- Not supported by Command Central.
- Platform Manager Syntax:

```
cc get configuration common  schema  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
schema	Required. Specifies the schema you want to retrieve. You can only retrieve schemas for common configuration types. The following list the schema names you can specify: ■ CommonSettings.xsd ■ EmailSettings.xsd ■ JDBCSettings.xsd ■ KeystoreTruststoreSettings.xsd ■ LicenseLocation.xsd ■ log4j.xsd ■ PortSettings.xsd
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Configuration types that have IDs that start with "COMMON-", for example COMMON-PORTS, are common configuration types that multiple products share. Common configuration types have normalized schemas that work for all products. However, these schemas still allow the following product-specific extensions:
  - You can have ExtendedProperties elements in the common schema XML files.
  - You can define common schema elements as optional.

  Each product maps a common schema to its specific use. To learn how a product supports common configuration types and how a product's configuration type is mapped to a common schema, use cc get configuration data to retrieve the data returned for a specific product's configuration instance. The structure of the configuration data can vary based on the run-time component, product that owns the run-time product, and in some cases also based on the specific instance of a configuration type.

Examples When Executing on Platform Manager

To execute a command on the Command Central server with host name "rubicon" and port "8090" to retrieve the "PortSettings.xsd" schema, using the authorization of the user with user name "Administrator" and password "manage":

```
cc get configuration common PortSettings.xsd --server http://rubicon2:8092/spm --username Adminisrator --password manage
```

cc get configuration compare

Compares the instances of a specified configuration type on two or more run-time components.

Syntax

- Command Central syntax:

```
cc get configuration compare configurationTypeId=typeid
runtimeComponentInfoId=id1 runtimeComponentInfoId=id2
[runtimeComponentInfoId=id3 ... runtimeComponentInfoId=idn] [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not supported by Platform Manager

Arguments and Options

Argument or Option	Description
`configurationTypeId=`*`typeid`*	Required. Specifies the configuration type to compare. You can determine the IDs for configuration types using cc list configuration types.

`runtimeComponentInfoId=`*`id1`*
`runtimeComponentInfoId=`*`id2`*
`[runtimeComponentInfoId=`*`id3`* `...`
`runtimeComponentInfoId=`*`idn`*`]`
	Required. Specifies the information IDs of two or more run-time components that you want to compare.
	
	The information ID is a combination of the alias name of the installation and the run-time component ID. For example, if the alias name of the installation is “sag1” and the run-time component ID is “OSGI-SPM”, the information ID is “sag1-OSGI-SPM”. You can view a list of installations and their alias names using cc list landscape nodes. You can determine the IDs for run-time components using cc list inventory components.

`[`*`options`*`]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The command returns the results of the comparison.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to compare the instances of the configuration type with ID “COMMON-PORTS” that are configured on the run-time component that have the information IDs “sag1-OSGI-SPM1” and “sag-OSGI-SPM2”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information displayed on the console in XML format:

```
cc get configuration compare configurationTypeId=COMMON-PORTS
runtimeComponentInfoId=sag1-OSGI-SPM1 runtimeComponentInfoId=sag1-OSGI-SPM2
--format xml --server http://rubicon:8090/cce --username Administrator
```

```
--password manage
```

cc create configuration data

Creates a new instance of a configuration type for a specified run-time component. For example, if you want to configure a new port for a Broker Server, use this command to supply the data for the configuration type COMMON-PORTS. By doing so, you create a new COMMON-PORTS instance.

Syntax

- Command Central syntax:

```
cc create configuration data  node_alias componentid typeid
{--input | -i} file{.xml|.json|.properties}  [options]
```

- Platform Manager syntax:

```
cc create configuration data  componentid typeid
{--input | -i} file{.xml|.json|.properties}  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--media-type | -m} content-type]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to create a new instance of a configuration type. You can determine the IDs for run-time components using cc list inventory components.

Argument or Option	Description
typeid	Required. Specifies the ID of the configuration type that identifies the type of instance you want to create. You can determine the IDs for configuration types using cc list configuration types. For information about the supported configuration types for a run-time component, see information in this reference for the product with which the run-time component is associated.
{--input | -i} *file*{.xml|.json|.properties}	Required. Identifies an input file that contains the configuration data. For more information, see "--input | -i" on page 42. **Note:** Based on the type of configuration instance you are attempting to create, all file types (.xml, .json, and .properties) might not be supported. Although not specifically supported, if you use plain text, the server attempts to convert the data into a supported format. **Tip:** To determine how to specify the data in the input file, use cc get configuration data to retrieve data for the same type of configuration instance you want to create. For example, if you want to use an XML file to specify the data to create an instance of a COMMON-PORTS configuration type, use cc get configuration data with the `--format xml` option to retrieve the data for an existing COMMON-PORTS instance in XML format.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Not all run-time components support the `cc create configuration data` command. For information about whether a run-time component supports a command, see information in this reference for the product with which the run-time component is associated.

- You can use cc get configuration common to validate input data that you want to use to create a configuration instance.
- You can retrieve schemas for common configuration types. You can use the schemas to validate an XML input data file. The schemas are available from a Platform Manager. For example, you might use the following to retrieve the log4j schema from a Platform Manager with host name “rubicon2” and port “8092”:

  http://rubicon:8092/spm/configuration/common/log4j.xsd

  You can also use cc get configuration common to retrieve schemas for common configuration types.
- The output from the `cc create configuration data` command includes:
  - Instance ID of the new configuration instance
  - Display name of the new instance
  - Description of the new configuration instance
  - ID of the associated configuration type
  - ID of the run-time component
  - URL of a physical configuration file if the data for the configuration instance is stored in a configuration file

Example When Executing on Command Central

The data to create a new instance of the COMMON-PORTS configuration type is in the c:\inputs\port_data.xml file. To create the new configuration instance for the run-time component with the ID “OSGI-SPM” that is installed in the installation with alias name “sag01”:

```
cc create configuration data sag01 OSGI-SPM COMMON-PORTS --input
c:\inputs\port_data.xml --password secret
```

Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Example When Executing on Platform Manager

The data to create a new instance of the COMMON-PORTS configuration type is in the c:\inputs\port_data.xml file. To create the new configuration instance for the run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”:

```
cc create configuration data OSGI-SPM COMMON-PORTS --input
c:\inputs\port_data.xml --server http://rubicon2:8092/spm
--password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Product-Specific Information

Configuration Types that webMethods Broker Supports

Configuration Types that IntegrationServer-instanceName Supports

Configuration Types that My webMethods Server-ENGINE Supports

Related Commands

cc delete configuration data

cc delete configuration data

cc get configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

cc list inventory components

cc delete configuration data

Deletes a configuration instance from a specified run-time component. For example, you might want to delete a previously created port on Integration Server.

Syntax

- Command Central syntax:

```
cc delete configuration data  node_alias componentid instanceid  [options]
```

- Platform Manager syntax:

```
cc delete configuration data  componentid instanceid  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[--force]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only.

Argument or Option	Description
	Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component from which you want to delete a configuration instance. You can determine the IDs for run-time components using cc list inventory components.
instanceid	Required. Specifies the ID of the instance you want to delete. You can determine the IDs for configuration instances using cc list configuration instances.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- You cannot delete some configuration data. In general, if you can create the configuration data, you can also usually delete it. For example, you can add a new Integration Server port, and you can also delete an Integration Server port. However, you cannot create new ports on Broker Server, and you cannot delete Broker Server ports. The command returns an error if you attempt to perform an unsupported operation.
- You can use cc exec configuration validation delete to determine whether it is valid to delete the configuration instance.

Example When Executing on Command Central

To delete the configuration instance with ID “COMMON-PORTS-com.softwareag.sshd.pid.properties” from the run-time component with ID “OSGI-SPM”, which is installed in the installation with alias name “sag01” using the authorization of the user with user name “Administrator” and password “manage”:

```
cc delete configuration data sag01 OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --username Administrator
--password manage
```

Because the `{--server | -s}` option is not specified, the command uses the default server. For more information, see "--server | -s" on page 50.

Example When Executing on Platform Manager

To delete the configuration instance with ID "COMMON-PORTS-com.softwareag.sshd.pid.properties" from the run-time component that has the ID "OSGI-SPM" and is managed by the Platform Manager with host name "rubicon2" and port "8092", and execute the command with the authorization of the user with user name "Administrator" and password "manage":

```
cc delete configuration data OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --server http://rubicon2:8092/spm
--username Administrator --password manage
```

Related Commands

cc create configuration data

cc create configuration data

cc get configuration data

cc update configuration data

cc list configuration instances

cc list inventory components

cc get configuration data

Retrieves data for a specified configuration instance that belongs to a specified run-time component. For example, you might want to retrieve the data for an instance of a configured port. The retrieved data for an instance of a port might include whether the port is enabled, the port number, and the port's protocol.

Syntax

- Command Central syntax:

```
cc get configuration data  node_alias componentid instanceid  [options]
```

- Platform Manager syntax:

```
cc get configuration data componentid instanceid  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {text | xml | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to retrieve instance data. You can determine the IDs for run-time components using cc list inventory components.
instanceid	Required. Specifies the ID of the instance for which you want to retrieve data. You can determine the IDs for configuration instances using cc list configuration instances.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Use cc get configuration instances or cc list configuration instances if you want information about an instance, such as the instance ID, the display name for an instance, and the description for an instance, rather than the data for an instance.
- If you do not specify the {--format | -f} option, the default output format is based on from where you execute the command:
  - If you execute the command from the command line, a batch script, or a shell script, the default format is plain text format.
  - If you execute the command from an Ant script, the default format is XML format.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve instance data for the configuration instance with ID “COMMON-

PORTS-com.softwareag.sshd.pid.properties" that belongs to the run-time component that has the ID "OSGI-SPM" and runs in the installation with alias name "sag01", using the authorization of the user with user name "Administrator" and password "manage", and have the information displayed on the console in XML format:

```
cc get configuration data sag01 OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --format xml
--server http://rubicon:8090/cce --username Administrator
--password manage
```

Example When Executing on Platform Manager

To retrieve instance data for the configuration instance with ID "COMMON-PORTS-com.softwareag.sshd.pid.properties" that belongs to the run-time component that has the ID "OSGI-SPM" and is managed by the Platform Manager with host name "rubicon2" and port "8092", and have the information returned to the output file "port_data" in XML format:

```
cc get configuration data OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --output port_data --format xml
--server http://rubicon2:8092/spm
```

Because the `{--username | -u}` and `{--password | -p}` options are not specified, the command uses the default user name and password. For more information, see "--username | -u" on page 52 and "--password | -p" on page 49.

Related Commands

cc create configuration data

cc delete configuration data

cc update configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

cc list inventory components

cc update configuration data

Updates the data for a specified configuration instance that belongs to a specified run-time component. For example, you might want to update the port number of a COMMON-PORTS configuration instance.

Syntax

- Command Central syntax:

```
cc update configuration data  node_alias componentid instanceid
{--input | -i} filename{.xml|.json|.properties}  [options]
```

- Platform Manager syntax:

```
   cc update configuration data  componentid instanceid
   {--input | -i} filename{.xml|.json|.properties}  [options]
options: [{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--media-type | -m} content-type]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component to which the instance you want to update belongs. You can determine the IDs for run-time components using cc list inventory components.
instanceid	Required. Specifies the ID of the instance for which you want to update data. You can determine the IDs for configuration instances using cc list configuration instances.
{--input | -i} *filename*{.xml| .json|.properties}	Required. Identifies an input file that contains the updated configuration data. For more information, see "--input | -i" on page 42. **Note:** Based on the type of configuration instance you are attempting to create, all file types (.xml, .json, and .properties) might not be supported. When updating instances of common configuration types, XML, json, and properties are all supported types of the input file. However, all of these file types might not

Argument or Option	Description
	be supported when creating instances of a product-specific configuration type. **Tip:** To determine how to specify the data in the input file, use cc get configuration data to retrieve data for the configuration instance you want to update. For example, if you want to use an XML file to specify the data to update an instance of a COMMON-PORTS configuration type, use cc get configuration data with the `--format xml` option to retrieve the data for the COMMON-PORTS instance in XML format.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The data in the input file must match the expected schema for the configuration type.
- You can use cc exec configuration validation update to validate input data that you want to use to update the configuration instance.
- You can retrieve schemas for common configuration types. You can use the schemas to validate an XML input data file. The schemas are available from a Platform Manager. For example, you might use the following to retrieve the log4j schema from a Platform Manager with host name "rubicon2" and port "8092":

  http://rubicon:8092/spm/configuration/common/log4j.xsd

  You can also us cc get configuration common to retrieve schemas for common configuration types.
- You can retrieve schemas for common configuration types. The schemas are available from a Platform Manager using the following where *hostname* is the host name of a Platform Manager server and *port* is its port number:

  http://*hostname* :*port* /spm/configuration/common/

  You can also us cc get configuration common to retrieve schemas for common configuration types.

Example When Executing on Command Central

The data to update a COMMON-PORTS instance is in the c:\inputs\port_data.xml file. To update the configuration instance with ID "COMMON-PORTS-com.softwareag.sshd.pid.properties" for the run-time component with ID "OSGI-SPM",

which is installed in the installation with alias name “sag01” using the authorization of the user with user name “Administrator” and password “manage”:

```
cc update configuration data sag01 OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --input c:\inputs\port_data.xml
--username Administrator --password manage
```

Because the `{--server | -s}` option is not specified, the command uses the default server. For more information, see "--server | -s" on page 50.

Example When Executing on Platform Manager

The data to update a COMMON-PORTS instance is in the c:\inputs\port_data.xml file. To update the configuration instance with ID “COMMON-PORTS-com.softwareag.sshd.pid.properties” for the run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”:

```
cc update configuration data OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --input c:\inputs\port_data.xml
--server http://rubicon2:8092/spm --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Product-Specific Information

Configuration Types that webMethods Broker Supports

Configuration Types that IntegrationServer-instanceName Supports

Configuration Types that My webMethods Server-ENGINE Supports

Related Commands

cc create configuration data

cc create configuration data

cc delete configuration data

cc get configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

cc list inventory components

cc get configuration instances

Retrieves information about a specific configuration instance that belongs to a specified run-time component. For example, you might want to retrieve information about an instance of a configuration properties file. Information about a configuration instance can include:

- Instance ID
- Type ID for the configuration type associated with the instance
- Display name for the instance
- Description of the instance
- URL providing the location of the configuration instance
- The ID of the run-time component to which the instance belongs

Syntax

- Command Central syntax:

```
cc get configuration instances node_alias componentid instanceid
[options]
```

- Platform Manager syntax:

```
cc get configuration instances componentid instanceid [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--quiet | -q}]
[{--password | -p} password]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to retrieve instance information.

Argument or Option	Description
	You can determine the IDs for run-time components using cc list inventory components.
instanceid	Required. Specifies the ID of the instance for which you want to retrieve information. You can determine the IDs for configuration instances using cc list configuration instances.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- To retrieve the data for a specific instance, use cc get configuration data.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information about the configuration instance with ID “COMMON-PORTS-com.softwareag.sshd.pid.properties” that belongs to the run-time component that has the ID “OSGI-SPM” and runs in the installation with alias name “sag01”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information displayed on the console in XML format:

```
cc get configuration instances sag01 OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --format xml
--server http://rubicon:8090/cce --username Administrator
--password manage
```

Example When Executing on Platform Manager

To retrieve information about the configuration instance with ID “COMMON-PORTS-com.softwareag.sshd.pid.properties” that belongs to the run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”, and have the information displayed on the console in XML format:

```
cc get configuration instances OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --format xml --server
http://rubicon2:8092/spm
```

Because the `{--username | -u}` and `{--password | -p}` options are not specified, the command uses the default user name and password. For more information, see "--username | -u" on page 52 and "--password | -p" on page 49.

Related Commands

cc get configuration data

cc list configuration instances

cc get configuration types

cc list configuration types

cc list inventory components

cc list configuration instances

Lists the configuration instances that belongs to a specified run-time component. Information about a configuration instance can include:

- Instance ID
- Type ID for the configuration type associated with the instance
- Display name for the instance
- Description of the instance
- URL providing the location of the configuration instance
- The ID of the run-time component to which the instance belongs

Syntax

- Command Central syntax:

```
cc list configuration instances node_alias componentid [instanceid]
[options]
```

- Platform Manager syntax:

```
cc list configuration instances componentid [instanceid] [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed.

Argument or Option	Description
	You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to list configuration instances. You can determine the IDs for run-time components using cc list inventory components.
[*instanceid*]	Optional. Specifies the ID of the instance for which you want to retrieve information. If you do not specify an instance ID, the command lists information for all instances that belong to the run-time component identified by the *componentid* argument.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- To retrieve the data for a specific instance, use cc get configuration data.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to list configuration instances that belong to the run-time component that has the ID “OSGI-SPM” and runs in the installation with alias name “sag01”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “config_instances” in XML format:

```
cc list configuration instances sag01 OSGI-SPM --format xml --output
config_instances --server http://rubicon:8090/cce
--username Administrator --password manage
```

Example When Executing on Platform Manager

To list configuration instances that belong to the run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in JavaScript Object Notation format:

```
cc list configuration instances OSGI-SPM --format json --server
http://rubicon2:8092/spm --username Administrator --password manage
```

Related Commands

cc get configuration data

cc get configuration instances

cc get configuration types

cc list configuration types

cc list inventory components

cc get configuration types

Retrieves information for a specified configuration type associated with a specified run-time component. Information about a configuration type can include:

- Type ID
- Display name if one is assigned; otherwise null
- Description if one is assigned; otherwise null
- Content type of the data

Syntax

- Command Central syntax:

```
cc get configuration types  node_alias componentid typeid  [options]
```

- Platform Manager syntax:

```
cc get configuration types  componentid typeid  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.

Argument or Option	Description
componentid	Required. Specifies the ID of the run-time component for which you want to retrieve a configuration type. You can determine the IDs for run-time components using cc list inventory components.
typeid	Required. Specifies the ID of the configuration type for which you want to retrieve information. You can determine the IDs for configuration types using cc list configuration types.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Run-time components support a set of configuration types. Use cc list configuration types to learn what configuration types that a run-time component supports.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information about the configuration type with ID “COMMON-PORTS” that is associated with run-time component that has the ID “OSGI-SPM” and runs in the installation with alias name “sag01”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information displayed on the console in XML format:

```
cc get configuration types sag01 OSGI-SPM COMMON-PORTS --format xml
--server http://rubicon:8090/cce --username Administrator
--password manage
```

Example When Executing on Platform Manager

To retrieve information about the configuration type with ID “COMMON-PORTS” that is associated with run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”, and have the output displayed on the console using the default format:

```
cc get configuration types OSGI-SPM COMMON-PORTS --server
http://rubicon2:8092/spm --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Product-Specific Information

Configuration Types that webMethods Broker Supports

Configuration Types that IntegrationServer-instanceName Supports

Configuration Types that My webMethods Server-ENGINE Supports

Related Commands

cc create configuration data

cc get configuration data

cc update configuration data

cc get configuration instances

cc list configuration instances

cc list configuration types

cc list inventory components

cc list configuration types

Lists information about configuration types for the specified run-time component. A run-time component can support both common configuration types, such as ports, logs, and licenses, and/or custom configuration types that are specific to the run-time component. Information about a configuration type can include:

- Type ID
- Display name if one is assigned; otherwise null
- Description if one is assigned; otherwise null
- Content type of the data

Syntax

- Command Central syntax:

```
cc list configuration types  node_alias componentid [typeid]  [options]
```

- Platform Manager syntax:

```
cc list configuration types  componentid [typeid]  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to list configuration types. You can determine the IDs for run-time components using cc list inventory components.
[*typeid*]	Optional. Specifies the ID of the configuration type for which you want to retrieve information. If you do not specify a type ID, the command lists information for all configuration types for the run-time component identified by the *componentid* argument.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Configuration types that have IDs that start with “COMMON-”, for example COMMON-PORTS, are common configuration types that multiple products share. Common configuration types have normalized schemas that work for all products. However, these schemas still allow product-specific extensions:
  - Having ExtendedProperties elements in the common schema XML files
  - Defining common schema elements as optional.

  Each product maps a common schema to its specific use. To learn how a product supports common configuration types and how a product’s configuration type is mapped to a common schemas, use cc get configuration data to retrieve the data returned for a specific product’s configuration instance. The structure of the configuration data can vary based on the run-time component, product that owns the run-time

product, and in some cases also based on the specific instance of a configuration type.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to list the configuration types for the run-time component that has the ID “OSGI-SPM” and is running in the installation with alias name “sag01”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “config_types” in XML format:

```
cc list configuration types sag01 OSGI-SPM --format xml
--output config_types --server http://rubicon:8090/cce
--username Administrator --password manage
```

Example When Executing on Platform Manager

To list the configuration types for the run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in JavaScript Object Notation format:

```
cc list configuration types OSGI-SPM --format json --server
http://rubicon2:8092/spm --username Administrator --password manage
```

Product-Specific Information

Configuration Types that webMethods Broker Supports

Configuration Types that IntegrationServer-instanceName Supports

Configuration Types that My webMethods Server-ENGINE Supports

Related Commands

cc create configuration data

cc get configuration data

cc update configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list inventory components

cc exec configuration validation create

Validates the configuration instance data in the supplied input file. If the input data is valid, you can then use cc create configuration data to create a configuration instance.

Syntax

- Command Central syntax:

```
cc exec configuration validation  node_alias componentid create typeid
{--input | -i} filename{.xml|.json|.properties}  [options]
```

- Platform Manager syntax:

```
cc exec configuration validation  componentid create typeid
{--input | -i} filename{.xml|.json|.properties}  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to validate instance data that you might want to use to create a new configuration type. You can determine the IDs for run-time components using cc list inventory components.
typeid	Required. Specifies the ID of the configuration type that identifies the type of instance data you want to validate. You can determine the IDs for configuration types using cc list configuration types. For information about the supported configuration types for a run-time component, see information in this reference for the product with which the run-time component is associated.

Argument or Option	Description
`{--input | -i}` *filename*`{.xml|.json|.properties}`	Required. Identifies an input file that contains the configuration data to validate. For more information, see "--input | -i" on page 42. **Note:** Based on the type of configuration data you are attempting to validate, all file types (.xml, .json, and .properties) might not be supported. Although not specifically supported, if you use plain text, the server attempts to convert the data into a supported format. **Tip:** To determine how to specify the data in the input file, use cc get configuration data to retrieve data for the same type of configuration instance you want to validate. For example, if you want to use an XML file for configuration data for a COMMON-PORTS configuration type, use cc get configuration data with the `--format xml` option to retrieve the data for an existing COMMON-PORTS instance in XML format.
`[`*options*`]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Not all run-time components support the `cc exec configuration validation create` command. For information about whether a run-time component supports a command, see information in this reference for the product with which the run-time component is associated.
- Use this command to determine whether data for a new configuration instance is valid. This command does not create a new configuration instance. If the data in the input file is valid, you create a new configuration instance using the data by executing cc create configuration data command and supplying the validated input file.
- The `cc exec configuration validation create` command outputs either no messages or informational, warning, and/or error messages.
  - When the command outputs no messages or only informational and warning messages, the input data is valid. You can use the data with the cc create configuration data command to create a configuration instance.
  - When the command outputs error messages, the input data is not valid. The cc create configuration data command will fail if you use the data to attempt to create a configuration instance.

Example When Executing on Command Central

The data for a COMMON-PORTS configuration type instance is in the c:\inputs \port_data.xml file. To validate the instance data for the run-time component with the ID “OSGI-SPM” that is installed in the installation with alias name “sag01”:

```
cc exec configuration validation sag01 OSGI-SPM create COMMON-PORTS
--input c:\inputs\port_data.xml --password secret
```

Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Example When Executing on Platform Manager

The data for a COMMON-PORTS configuration type instance is in the c:\inputs \port_data.xml file. To validate the instance data for the run-time component with the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”:

```
cc exec configuration validation OSGI-SPM create COMMON-PORTS
--input c:\inputs\port_data.xml --server http://rubicon2:8092/spm
--password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Product-Specific Information

Configuration Types that IntegrationServer-instanceName Supports

Configuration Types that My webMethods Server-ENGINE Supports

Related Commands

cc create configuration data

cc exec configuration validation delete

cc exec configuration validation delete

cc exec configuration validation delete

Determines whether a configuration instance can be deleted. If check is successful, you can then use cc delete configuration data to delete the configuration instance.

Syntax

- Command Central syntax:

  ```
  cc exec configuration validation  node_alias componentid delete instanceid
  [options]
  ```

- Platform Manager syntax:

```
   cc exec configuration validation  componentid delete instanceid  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component that owns the instance. You can determine the IDs for run-time components using cc list inventory components.
instanceid	Required. Specifies the ID of the instance you want to check. You can determine the IDs for configuration instances using cc list configuration instances.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Use this command to determine whether you can delete a configuration instance. This command does not delete the configuration instance.
- The `cc exec configuration validation delete` command outputs either no messages or informational, warning, and/or error messages.

- When the command outputs no messages or only informational and warning messages, the check is successful. You can use the cc delete configuration data command to delete the configuration instance.
- When the command outputs error messages, the check failed. The cc delete configuration data command will fail if attempt to delete the configuration instance.

Example When Executing on Command Central

To check whether you can delete the configuration instance with ID "COMMON-PORTS-com.softwareag.sshd.pid.properties" from the run-time component with ID "OSGI-SPM", which is installed in the installation with alias name "sag01" using the authorization of the user with user name "Administrator" and password "manage":

```
cc delete configuration data sag01 OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties --username Administrator
--password manage
```

Because the `{--server | -s}` option is not specified, the command uses the default server. For more information, see "--server | -s" on page 50.

Example When Executing on Platform Manager

To check whether you can delete the configuration instance with ID "COMMON-PORTS-com.softwareag.sshd.pid.properties" from the run-time component that has the ID "OSGI-SPM" and is managed by the Platform Manager with host name "rubicon2" and port "8092", and execute the command with the authorization of the user with user name "Administrator" and password "manage":

```
cc delete configuration data OSGI-SPM COMMON-PORTS-
com.softwareag.sshd.pid.properties -server http://rubicon2:8092/spm
--username Administrator --password manage
```

Related Commands

cc create configuration data

cc exec configuration validation delete

cc exec configuration validation delete

cc exec configuration validation update

Validates the configuration instance data in the supplied input file to determine whether you can use it to update a specified configuration instance. If the input data is valid, you can then use cc delete configuration data to update the configuration instance.

Syntax

- Command Central syntax:

  ```
  cc exec configuration validation  node_alias componentid update instanceid
  {--input | -i} filename{.xml|.json|.properties} [options]
  ```

- Platform Manager syntax:

```
   cc exec configuration validation  componentid update instanceid
   {--input | -i} filename{.xml|.json|.properties}  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to validate instance data that you might want to use to update a configuration instance. You can determine the IDs for run-time components using cc list inventory components.
instanceid	Required. Specifies the ID of the instance. You can determine the IDs for configuration instances using cc list configuration instances.
{--input | -i} *filename*{.xml|.json|.properties}	Required. Identifies an input file that contains the configuration data to validate. For more information, see "--input | -i" on page 42. **Note:** Based on the type of configuration data you are attempting to validate, all file types (.xml, .json, and .properties) might not be supported. Although not specifically supported, if you use plain text, the server attempts to convert the data into a supported format. **Tip:** To determine how to specify the data in the input file, use cc get configuration data to retrieve data for the same type of configuration instance you want

Argument or Option	Description
	to validate. For example, if you want to use an XML file for configuration data for a COMMON-PORTS configuration type, use cc get configuration data with the `--format xml` option to retrieve the data for an existing COMMON-PORTS instance in XML format.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Use this command to determine whether data to update a configuration instance is valid. This command does not update the configuration instance. If the data in the input file is valid, you update the configuration instance using the data by executing cc delete configuration data command and supplying the validated input file.
- The `cc exec configuration validation update` command outputs either no messages or informational, warning, and/or error messages.
  - When the command outputs no messages or only informational and warning messages, the input data is valid. You can use the data with the cc delete configuration data command to update the configuration instance.
  - When the command outputs error messages, the input data is not valid. The cc delete configuration data command will fail if you use the data to attempt to update the configuration instance.

Example When Executing on Command Central

The data to update a COMMON-PORTS configuration type instance is in the c:\inputs\port_data.xml file. To validate the data for the configuration instance with ID "COMMON-PORTS-com.softwareag.sshd.pid.properties" for the run-time component with ID "OSGI-SPM", which is installed in the installation with alias name "sag01" using the authorization of the user with user name "Administrator" and password "manage":

```
cc exec configuration validation sag01 OSGI-SPM update COMMON-PORTS-
com.softwareag.sshd.pid.properties --input c:\inputs\port_data.xml
--username Administrator --password manage
```

Because the `{--server | -s}` option is not specified, the command uses the default server. For more information, see "--server | -s" on page 50.

Example When Executing on Platform Manager

The data to update a COMMON-PORTS instance is in the c:\inputs\port_data.xml file. To validate the data for the configuration instance with ID "COMMON-PORTS-com.softwareag.sshd.pid.properties" for the run-time component that has the ID "OSGI-

SPM" and is managed by the Platform Manager with host name "rubicon2" and port "8092":

```
cc exec configuration validation OSGI-SPM update COMMON-PORTS-
com.softwareag.sshd.pid.properties --input c:\inputs\port_data.xml
--server http://rubicon2:8092/spm --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies "secret" for the user's password.

Product-Specific Information

Configuration Types that IntegrationServer-instanceName Supports

Configuration Types that My webMethods Server-ENGINE Supports

Related Commands

cc create configuration data

cc exec configuration validation delete

cc exec configuration validation delete

5 Diagnostics Logs Commands

cc get diagnostics logs

Retrieves log entries from a log file. Log information includes the date, time, and description of events that occurred with a specified run-time component.

Syntax

- Command Central syntax:

  You can optionally identify log(s) by supplying either a regular expression or search text.

  - To optionally specify a regular expression:

    ```
    cc get  diagnostics logs  node_alias runtime_componentid logid
    {full | tail | head} [lines=number] [(regex=expression]
    [options]
    ```

  - To optionally specify search text:

    ```
    cc get  diagnostics logs  node_alias runtime_componentid logid
    {full | tail | head} [lines=number] [search=text]  [options]
    ```

- Platform Manager syntax:

  You can optionally identify log(s) by supplying either a regular expression or search text.

  - To optionally specify a regular expression:

    ```
    cc get  diagnostics logs  runtime_componentid logid
    {full | tail | head} [lines=number] [(regex=expression] [options]
    ```

  - To optionally specify search text:

    ```
    cc get  diagnostics logs runtime_componentid logid
    {full | tail | head} [lines=number] | head [lines=number]}
    [search=text]  [options]

     options:
    [{--debug | -d}]
    [{--error | -r} file]
    [{--log | -l} file]
    [{--output | -o} file]
    [{--password | -p} password]
    [{--quiet | -q}]
    [{--server | -s} url]
    [{--username | -u} user_name]
    ```

Arguments and Options

Argument or Option	Description
node_alias	Required. Only for Command Central. Specifies the alias name of the Platform Manager that

Argument or Option	Description
	manages the run-time component for which you want to retrieve information. You can determine the alias name of the Platform Manager node by using cc list landscape nodes
runtime_componentid	Required. Specifies the ID of the run-time component for which you want to retrieve information. You can determine the IDs for run-time components using cc list inventory components.
logid	Required. Specifies the ID of the log for which you want to retrieve information. You can determine the IDs for logs using cc list diagnostics logs.
{full | tail | head}	Required. Identifies the log entries you want to retrieve. ■ Specify full to retrieve all log entries. ■ Specify tail to retrieve the most recent log entries from the end of the log file. ■ Specify head to retrieve entries from the beginning of the log file.
[lines=*number*]	Optional. Use only with the tail or head parameters. Specifies the number of log entries to return. If you omit lines=*number* the command returns 100 entries.
[regex=*expression*]	Optional. Narrows the retrieved log entries to those that meet the specified regular expression.
[search=*text*]	Optional. Narrows the retrieved log entries to those that contain the specified search text.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- By default, the command returns log entries or the full log in plain text format. If you specify a zip file for the output format, the command returns the full log file in a zip archive.
- Specify either `regex` or `search`. If you specify both, the command narrows the result using the regular expression you specify with `regex` and ignores the search text you specify with `search`.
- If you use `regex` to specify a regular expression or `search` to specify search text, and the regular expression or search text identify no log entries, the command returns no results.
- When you use `lines` with `regex` or `search`, the command returns the specified number of lines in the log that contain the specified regular expression or text. When you use `lines` without `regex` or `search`, the command returns the specified number of lines from the top or bottom of the log. For example:
  - `tail lines=10 search=JMX` returns up to ten log entries with the word "JMX".
  - `tail lines=10` returns the last ten log entries.
- When you use `{full | tail | head}` with large log files, include the `-o file` option to specify an output file. Writing a large number of log entries to the console may result in an out of memory errors.

  For more information about the `-o file` option, see --output | -o.
- To avoid performance issues, do not specify a large number for `lines` when using with `tail`, `search` or `regex`.

Examples When Executing on Command Central

- The run-time component with ID "OSGI-SPM" is managed by the Platform Manager registered as "is-dev". The run-time component has a log with ID "default.log". Use the following command to filter the log to entries that contain "JMX" as a word or part of a word, and return up to 20 matching entries. The results are written to the console.

  ```
  cc get diagnostics logs is-dev OSGI-SPM default.log tail lines=20
  regex=.*JMX.*
  ```
- The run-time component with ID "OSGI-SPM" is managed by the Platform Manager registered as "is-dev". The run-time component has a log with ID "default.log". Use the following command to filter the log to entries that contain the word "JMX", and return up to 20 matching entries. The results are written to the console.

  ```
  cc get diagnostics logs is-dev OSGI-SPM default.log head lines=20
  search=JMX --password secret
  ```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies "secret" for the user's password.

Examples When Executing on Platform Manager

- The run-time component with ID “OSGI-SPM” is managed by the Platform Manager with host name “rubicon2” and port “8092”. The run-time component has a log with ID “default.log”. Use the following command to filter the log to entries that contain “JMX” as a word or part of a word, and return up to 20 matching entries. The results are written to the console.

  ```
  cc get diagnostics logs OSGI-SPM default.log tail lines=20
  regex=.*JMX.* --server http://rubicon2:8092/spm --password secret
  ```

- The run-time component with ID “OSGI-SPM” is managed by the Platform Manager with host name “rubicon2” and port “8092”. The run-time component has a log with ID “default.log”. Use the following command to filter the log to entries that contain the word “JMX”, and return up to 20 matching entries. The results are written to the console.

  ```
  cc get diagnostics logs OSGI-SPM default.log head lines=20
  search=JMX --server http://rubicon2:8092/spm --password secret
  ```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Related Commands

cc list diagnostics logs

cc list inventory components

cc get diagnostic logs export file

Exports one or more log files for a specified run-time component in a zip archive file.

Syntax

- Command Central syntax:
  - To export log(s) for a specified run-time component:

    ```
    cc get diagnostics logs node_alias runtime_componentid
    [logid+logid...] export -o file [options]
    ```

  - To export all available logs for a specified run-time component:

    ```
    cc get diagnostics logs node_alias runtime_componentid
    export -o file [options]
    ```

  - To export logs for all run-time components:

    ```
    cc get diagnostics logs node_alias export -o file [options]
    ```

- Not supported by Platform Manager.

```
 options:
[{--debug | -d}]
```

```
[{--error | -r} file]
[{--format | -f} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Required. Specifies the alias name of the Platform Manager that manages the run-time component for which you want to retrieve information.
runtime_componentid	Required. Specifies the ID of the run-time component for which you want to retrieve information. You can determine the IDs for run-time components using cc list inventory components.
[*logid+logid...*]	A list of IDs for the log(s) that you want to export. Use the + sign as separator. You can determine the IDs for logs using cc list diagnostics logs.
-o *file*	Required. Specifies the name of the output file. If the file you specify does not exist, the command creates it. For more information about the output file command, see "--output | -o" on page 47.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- The run-time component with ID “OSGI-SPM” is managed by the Platform Manager registered as “is-dev”. The run-time component has logs with IDs “error.log” and “default.log”. Use the following command to export the two logs to a zip archive file with name “test.zip”.

  ```
  cc get diagnostics logs  is-dev OSGI-SPM error.log+default.log export
  -o test.zip
  ```

- The run-time component with ID “OSGI-SPM” is managed by the Platform Manager registered as “is-dev”. Use the following command to export all available logs for the “OSGI-SPM” component to a zip archive file with name “test.zip”.

```
cc get diagnostics logs  is-dev OSGI-SPM export -o test.zip
```

cc list diagnostics logs

Lists the log files that a specified run-time component supports. Information for log files includes:

- Location of the log file
- Log ID for the log file
- Date the log file was last modified
- Size of the log file

This command returns information about the log files rather than the contents of the logs. To retrieve the contents of the log, use cc get diagnostics logs.

Syntax

- Command Central syntax:

```
cc list diagnostics logs  node_alias  runtime_componentid  [logid]
[options]
```

- Platform Manager syntax:

```
cc list diagnostics logs  runtime_componentid  [logid]  [options]

 options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {xml | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Required. Only for Command Central. Specifies the alias name of the Platform Manager that manages the run-time component for which you want to retrieve information.

Argument or Option	Description
runtime_componentid	Required. Specifies the ID of the run-time component for which you want to retrieve information. You can determine the IDs for run-time components using cc list inventory components.
[*logid*]	Optional. Specifies the ID of the log for which you want to retrieve information.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- If you do not specify the `{--format | -f}` option, the default output format is tab-separated values text.

Examples When Executing on Command Central

- To list all log files for the run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager registered as “is-dev”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “loginfo”:

```
cc list diagnostics logs is-dev OSGI-SPM --output
loginfo --username Administrator --password manage
```

- To list information for the log file with ID “default.log” from the run-time component that has ID “OSGI-SPM” and is managed by the Platform Manager registered as “is-dev”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in JavaScript Object Notation format:

```
cc list diagnostics logs is-dev OSGI-SPM default.log
--format json --username Administrator --password manage
```

Examples When Executing on Platform Manager

- To list all log files for the run-time component that has the ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “loginfo”:

```
cc list diagnostics logs OSGI-SPM --server http://rubicon2:8092/spm
--output loginfo --username Administrator --password manage
```

- To list information for the log file with ID “123124” from the run-time component that has ID “OSGI-SPM” and is managed by the Platform Manager with host name

"rubicon2" and port "8092", using the authorization of the user with user name "Administrator" and password "manage", and have the information returned to the console in JavaScript Object Notation format:

```
cc list diagnostics logs OSGI-SPM 123124 --server
http://rubicon2:8092/spm --format json --username Administrator
--password manage
```

Related Commands

cc get diagnostics logs

6 Inventory Commands

cc get inventory components

Retrieves information about a specified run-time component. Information about a run-time component can include:

- Display name
- ID for the run-time component
- ID of the product to which this run-time component belongs
- Run-time component category, which can be one of the following:
  - PROCESS for run-time components that functions on its own. These are referred to as *instances* in the Web user interface.
  - ENGINE for run-time components that cannot function on their own, but rather run within a PROCESS run-time component. These are referred to as *components* in the Web user interface.

Syntax

- Command Central syntax:

```
cc get inventory components  node_alias componentid  [options]
```

- Platform Manager syntax:

```
cc get inventory components  componentid  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation for which you want to retrieve component information. You can view a list of installations and their aliases using cc list landscape nodes.

Argument or Option	Description
componentid	Required. Specifies the ID of the run-time component for which you want to retrieve information. You can determine the IDs for run-time components using cc list inventory components.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information for the run-time component that has the component ID “OSGI-SPM” and is installed on the installation with the alias name “sag01”, and have the output returned to the console in JavaScript Object Notation format:

```
cc get inventory components sag01 OSGI-SPM --server http://rubicon:8090/cce
--format json --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Example When Executing on Platform Manager

To retrieve information for the run-time component that has the component ID “OSGI-SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information displayed on the console in XML format:

```
cc get inventory components OSGI-SPM --server http://rubicon2:8092/spm
--format xml --username Administrator --password manage
```

Related Commands

cc list inventory components

cc update inventory components

cc list inventory components

Lists information about run-time components. Information about a run-time component can include:

- Display name

- ID for the run-time component
- ID of the product to which this run-time component belongs
- Run-time component category, which can be one of the following:
  - PROCESS for run-time components that functions on its own. These are referred to as *instances* in the Web user interface.
  - ENGINE for run-time components that cannot function on their own, but rather run within a PROCESS run-time component. These are referred to as *components* in the Web user interface.

Syntax

- Command Central syntax:
  - To list components for a specified installation:

    ```
    cc list inventory components  [node_alias] [componentid]
    [options]
    ```

  - To list components that match specified search criteria:

    ```
    cc list inventory components  [criteria] [start=number]
    [size=number] [options]
    ```

- Platform Manager syntax:

```
   cc list inventory components  [componentid]  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
[node_alias]	Command Central only. Optional. Specifies the alias name of the installation for which you want to retrieve component information. You can view a list of installations and their aliases using cc list landscape nodes. If you do not specify an alias name nor search criteria, the command lists information for all

Argument or Option	Description
	run-time components for all installations that Command Central manages.
`[componentid]`	Optional. Specifies the ID of the run-time component for which you want to retrieve information. You can determine the IDs for run-time components using cc list inventory components.
`[criteria]`	Command Central only. Optional. Narrows down the list of returned run-time components to only those that match the search criteria you specify. For more information, see "Specifying Search Criteria for Inventory Commands" on page 103.
`[start=number]`	Command Central only. Optional. Limits the results the command returns those starting at specified number in the results. For example, if you want to return information for the 5th through the 8th run-time components in the results, use `start=5 size=4`.
`[size=number]`	Command Central only. Optional. Limits the number of results you want returned. For example, if you specify `size=1`, the command returns information for only one run-time component.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- To list all run-time components that the Command Central with host name “rubicon” and port “8090” manages, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “components” in XML format:

```
cc list inventory components --format xml --output components
--server http://rubicon:8090/cce --username Administrator
--password manage
```

- To list the same run-time components as the first example above, but restrict the number of returned run-time components to only 5:

  ```
  cc list inventory components size=5 --format xml --output components
  --server http://rubicon:8090/cce --username Administrator
  --password manage
  ```

- To list the 10th through the 15th run-time components in the results and return the output to the console in XML format:

  ```
  cc list inventory components start=10 size=6 --format xml
  --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

- To list run-time components and use search criteria to narrow the results to only those that are installed in the installation with alias name “sag01” and that have the component ID “OSGI-CCE” and return the output to the console in JavaScript Object Notation format:

  ```
  cc list inventory components nodeAlias=sag01
  runtimeComponentId=OSGI-CCE --format json --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

- To list run-time components and use search criteria to narrow the results to only those that are installed in the installation with alias names “sag01” or “sag03” and return the output to the console in xml format:

  ```
  cc list inventory components logicalOperator=OR nodeAlias=sag01
  nodeAlias=sag03 --format xml --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Example When Executing on Platform Manager

To list all run-time components managed by the Platform Manager with host name “rubicon2” and port “8092”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in XML format:

```
cc list inventory components --format xml
--server http://rubicon2:8092/spm --username Administrator
--password manage
```

Related Commands

cc get inventory components

cc update inventory components

Specifying Search Criteria for Inventory Commands

When using the `cc list inventory components` or `cc list inventory products` commands to list run-time components or products, you can specify search criteria to narrow the results that the command returns. Supply the search criteria using the following format:

property1=value1 property2=value2 ...

For the search criteria, you specify property values to match, for example `runtimeComponentId=OSGI-CCE`, where `runtimeComponentId` is the property and the value to match is `OSGI-CCE`.

Property Names You Can Use in the Search Criteria

Command	Property Names
`cc list inventory components`	■ `nodeName` ■ `nodeAlias` ■ `nodeUrl` ■ `environmentName` ■ `environmentAlias` ■ `runtimeComponentInfoId` ■ `runtimeComponentId` ■ `runtimeComponentDisplayName` ■ `runtimeComponentProductId` ■ `runtimeComponentCategory` ■ `runtimeComponentRuntimeStatus` ■ `runtimeComponentRuntimeParentId`
`cc list inventory fixes`	■ `nodeName` ■ `nodeAlias` ■ `nodeUrl` ■ `environmentName` ■ `environmentAlias` ■ `fixId` ■ `fixDisplayName` ■ `fixVersion` ■ `fixGroup` ■ `fixProducts`
`cc list inventory products`	■ `nodeName` ■ `nodeAlias`

Command	Property Names
	■ nodeUrl ■ environmentName ■ environmentAlias ■ productId ■ productCanonicalId ■ productDisplayName ■ productParentId ■ productGroup ■ productProfileDir ■ productCode ■ productVersion ■ productInstallTime

Specifying the Value

When specifying the value, you can include the * pattern-matching character to match multiple characters. For example, if you want to narrow the list of returned products to only those with "mws" anywhere in their product display names, use the following search criterion:

```
productDisplayName=*mws*
```

Important: The search is case-sensitive.

Logical Operators Used When Specifying Multiple Search Properties

If you specify multiple search items, by default, the command performs an AND operation to return results that match all the specified criteria. For example, to narrow the list of returned products to those with "mws" anywhere in their product display names and that are version 9.0 or later, use the following search criteria:

```
productDisplayName=*mws* productVersion=9.0*
```

You can use an OR operation with two properties. To do so, specify the `logicalOperator=OR` argument. For example, to narrow the list of returned run-time components to those installed in installations that have the alias name "sag01" or "sag02", use the following search criteria:

```
nodeAlias=sag01 logicalOperator=OR nodeAlias=sag02
```

Related Commands

cc list inventory components

cc list inventory products

cc list inventory fixes

cc update inventory components

Updates the display name and/or icon associated with a specified run-time component.

Syntax

- Command Central syntax:

```
cc update inventory components  node_alias componentid
{--input | -i} filename{.xml|.json|.properties}  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not supported on Platform Manager

Arguments and Options

Argument or Option	Description
node_alias	Required. Specifies the alias name of the installation for which you want to retrieve component information. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component that you want to update. You can determine the IDs for run-time components using cc list inventory components.
{--input | -i} *filename*{.xml|.json|.properties}	Required. Identifies an input file that contains the updated data for the run-time component. For more information, see "--input | -i" on page 42.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The information that you can update for a run-time component is the display name and icon.
- To update the icon for a run-time component, you supply the icon ID of the new icon. To determine the icon IDs of installed icons, use the cc list resources icons command.

Example When Executing on Command Central

To update the run-time component with ID OSGI-SPM that is installed in the installation with alias name "sag01" using the data in the c:\inputs\component_data.xml file:

```
cc update inventory components sag01 OSGI-SPM
--input c:\inputs\component_data.xml
```

Because the `{--server | -s}`, `{--username | -u}`, and `{--password | -p}` options are not specified, the command uses the default server, user name, and password. For more information, see "--server | -s" on page 50, "--username | -u" on page 52, and "--password | -p" on page 49.

Related Commands

cc get inventory components

cc list inventory components

cc list resources icons

cc get inventory fixes compare

Compares the fixes installed in two or more installations.

Syntax

- Command Central syntax:

```
cc get inventory fixes compare nodeAlias=alias1 nodeAlias=alias2
[nodeAlias=alias3 ...nodeAlias=aliasn] [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not supported by Platform Manager

Arguments and Options

Argument or Option	Description
`nodeAlias=`*`alias1`* `nodeAlias=`*`alias2`* `[nodeAlias=`*`alias3`* `...` *`nodeAlias=aliasn`*`]`	Required. Specifies the alias names of two or more installations that you want to compare. You can view a list of installations and their aliases using cc list landscape nodes.
`[`*`options`*`]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The command returns the results of the comparison.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to compare the fixes applied to the installations with alias names “sag01” and “sag02” and have the output returned to the console in XML format:

```
cc get inventory fixes compare nodeAlias=sag01 nodeAlias=sag02
--server http://rubicon:8090/cce --format xml --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Related Commands

cc list inventory fixes

cc list inventory fixes

Lists information about fixes that have been applied to products. Information about fixes can include:

- Fix ID
- Fix name
- Version of the fix
- Product to which the fix is applied

Syntax

- Command Central syntax:
  - To list information for a specified fix:

    ```
    cc list inventory fixes  [fixid]  [options]
    ```

  - To list fixes that match specified search criteria:

    ```
    cc list inventory fixes  [criteria] [start=number] [size=number]
    [options]
    ```

- Platform Manager syntax:

```
   cc list inventory fixes  [fixid]  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--quiet | -q}]
[{--password | -p} password]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
fixid	Optional. Specifies the ID of the fix for which you want information.
[*criteria*]	Command Central only. Optional. Narrows down the list of returned fixes to only those that match the search criteria you specify. For more information, see "Specifying Search Criteria for Inventory Commands" on page 103.
[start=*number*]	Command Central only. Optional. Limits the results the command returns those starting at specified number in the results. For example, if you want to return information for the 5th through the 8th products in the results, use `start=5 size=4`.
[size=*number*]	Command Central only.

Argument or Option	Description
	Optional. Limits the number of results you want returned. For example, if you specify `size=1`, the command returns information for only one product.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information for the fix with ID “CCE_9-0_Fix2” and have the output returned to the console in JavaScript Object Notation format:

  ```
  cc get inventory fixes CCE_9-0_Fix2 --server http://rubicon:8090/cce
  --format json --password secret
  ```

  Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

- To list the fixes applied to all products that the Command Central with host name “rubicon” and port “8090” manages, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “fixlist” in XML format:

  ```
  cc list inventory fixes --format xml --output fixlist
  --server http://rubicon:8090/cce --username Administrator
  --password manage
  ```

- To list the same fixes as the first example above, but restrict the number of returned products to only 5:

  ```
  cc list inventory fixes size=5 --format xml --output productlist
  --server http://rubicon:8090/cce --username Administrator
  --password manage
  ```

- To list the fixes that are version 9.0 or later and also contain “wMFix” in their fix IDs, and return the output to the console in JavaScript Object Notation format:

  ```
  cc list inventory fixes fixversion=9.0* fixId=*wMFix* --format json
  --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

- To list the 10th through the 15th fixes in the results and return the output to the console in XML format:

  ```
  cc list inventory fixes start=10 size=6 --format xml --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies "secret" for the user's password.

Example When Executing on Platform Manager

To list information about the fixes applied to all the products that are managed by the Platform Manager with host name "rubicon2" and port "8092", using the authorization of the user with user name "Administrator" and password "manage", and have the information returned to the console in XML format:

```
cc list inventory fixes --format xml --username Administrator
--password manage --server http://rubicon2:8092/spm
```

Related Commands

cc get inventory fixes compare

cc get inventory products

Retrieves information about a specified product. Information about a product can include:

- Display name
- ID for the product
- Product code
- Product version
- Date and time the product was installed

Syntax

- Command Central syntax:

  ```
  cc get inventory products  node_alias productid  [options]
  ```

- Platform Manager syntax:

  ```
  cc get inventory products  productid  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
  ```

Arguments and Options

Argument or Option	Description
`[node_alias]`	Command Central only. Required. Specifies the alias name of the installation for which you want to retrieve product information. If you do not specify an alias name, the command lists all products in all installations that Command Central manages. You can view a list of installations and their aliases using cc list landscape nodes.
`productid`	Required. Specifies the product ID of the product for which you want to retrieve product information. You can determine the IDs for products using cc list inventory products.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information for the product that has the ID “SPM” and is installed on the installation with the alias name “sag01”, and have the output returned to the console in JavaScript Object Notation format:

```
cc get inventory products sag01 SPM --server http://rubicon:8090/cce
--format json --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Example When Executing on Platform Manager

To retrieve information for the product that has the ID “SPM” and is managed by the Platform Manager with host name “rubicon2” and port “8092”, and have the output returned to the console in XML format:

```
cc get inventory products SPM --server http://rubicon2:8092/spm
--format xml --password secret
```

Because the {--username | -u} option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies "secret" for the user's password.

Related Commands

cc get inventory products compare

cc list inventory products

cc get inventory products compare

Compares the products installed in two or more installations.

Syntax

- Command Central syntax:

```
cc get inventory products compare nodeAlias=alias1 nodeAlias=alias2
[nodeAlias=alias3 ... nodeAlias=aliasn] [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not supported by Platform Manager

Arguments and Options

Argument or Option	Description
nodeAlias=*alias1* nodeAlias=*alias2* [nodeAlias=*alias3* ... *nodeAlias=aliasn*]	Required. Specifies the alias names of two or more installations that you want to compare. You can view a list of installations and their aliases using cc list landscape nodes.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The command returns the results of the comparison.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to compare the products installed in the installations with alias names “sag01” and “sag02” and have the output returned to the console in XML format:

```
cc get inventory products compare nodeAlias=sag01 nodeAlias=sag02
--server http://rubicon:8090/cce --format xml --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Related Commands

cc get inventory products

cc list inventory products

cc list inventory products

Lists information about products. Information about a product can include:

- Display name
- ID for the product
- Product code
- Product version
- Date and time the product was installed

Syntax

- Command Central syntax:
  - To list products for a specified installation:

    ```
    cc list inventory products  [node_alias] [productid] [options]
    ```

  - To list products that match specified search criteria:

    ```
    cc list inventory products  [criteria] [start=number]
    [size=number] [options]
    ```

- Platform Manager syntax:

```
   cc list inventory products  [productid]  [options]
options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
```

```
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
[*node_alias*]	Command Central only. Optional. Specifies the alias name of the installation for which you want to retrieve product information. You can view a list of installations and their aliases using cc list landscape nodes. If you do not specify an alias name nor search criteria, the command lists all products in all installations that Command Central manages.
[*productid*]	Optional. Specifies the ID of the product for which you want to retrieve information.
[*criteria*]	Command Central only. Optional. Narrows down the list of returned products to only those that match the search criteria you specify. For more information, see "Specifying Search Criteria for Inventory Commands" on page 103.
[start=*number*]	Command Central only. Optional. Limits the results the command returns those starting at specified number in the results. For example, if you want to return information for the 5th through the 8th products in the results, use `start=5 size=4`.
[size=*number*]	Command Central only. Optional. Limits the number of results you want returned. For example, if you specify `size=1`, the command returns information for only one product.

Argument or Option	Description
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- To list all products that the Command Central with host name “rubicon” and port “8090” manages, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “productlist” in XML format:

  ```
  cc list inventory products --format xml --output productlist
  --server http://rubicon:8090/cce --username Administrator
  --password manage
  ```

- To list the same products as the first example above, but restrict the number of returned products to only 5:

  ```
  cc list inventory products size=5 --format xml --output productlist
  --server http://rubicon:8090/cce --username Administrator
  --password manage
  ```

- To list products that are version 9.0 or later and also contain “Platform” in their display name and return the output to the console in JavaScript Object Notation format:

  ```
  cc list inventory products productVersion=9.0*
  productDisplayName=*Platform* --format json --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

- To list the 10th through the 15th products in the results and return the output to the console in XML format:

  ```
  cc list inventory products start=10 size=6 --format xml
  --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Example When Executing on Platform Manager

To list information about the products that are managed by the Platform Manager with host name “rubicon2” and port “8092”, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in XML format:

```
cc list inventory products --format xml --username Administrator
```

```
--password manage --server http://rubicon2:8092/spm
```

Related Commands

cc get inventory products compare

cc get inventory products

7 Instance Management Commands

cc create instances

Creates a new instance of an installed product.

Syntax

- Command Central syntax:

```
cc create instances  node_alias product
[key=value] | [-i file{.xml|.json|.properties}]  [options]
```

- Platform Manager syntax:

```
cc create instances  product
[key=value] | [-i file{.xml|.json|.properties}]  [options]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which to create the product instance. You can view a list of installations and their aliases using cc list landscape nodes.
product	Required. Specifies the product ID of the installed product or run-time component for which you want to create a new instance. Valid values for this option are only the product IDs included in the list of products returned from the `cc list instances` *node_alias* `supportedproducts` command.
[*key=value*]	Optional. A list of properties that describe the elements of the new instance, such as name and port settings. The properties included in this list are product specific.
[-i [*file*{.xml|.json|.properties}]	Optional. Identifies an input file that contains the product instance data. For

Argument or Option	Description
	more information, see "--input	-i" on page 42. For the correct format of an XML properties file, see the Properties class in the Oracle Java Platform Standard Edition API specification.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

The command returns job information that you can monitor using the cc list job manager command.

Example When Executing on Command Central

- To create the new instance for an installed Integration Server with instance name “is-instance2”, port “5560”, diagnostic port “8083”, and JMX port “10058” in the installation with alias name “productionNode2”:

```
cc create instances productionNode2 integrationServer
instance.name=is-instance2 http.port=5560 diagnostic.port=8083
jmx.port=10058
```

Examples When Executing on Platform Manager

- To create the new instance for an installed Integration Server with instance name “is-instance2”, port “5560”, diagnostic port “8083”, and JMX port “10058”:

```
cc create instances integrationServer instance.name=is-instance2
http.port=5560 diagnostic.port=8083 jmx.port=10058
```

- To create the new instance for an installed Integration Server using the instance data in the instance-settings.properties file, located it in the current directory:

```
cc create instances integrationServer -i instance-settings.properties
```

- To create the new instance for an installed Integration Server using the instance data in the instance.settings.xml file, located in the current directory:

```
cc create instances integrationServer -i instance-settings.xml
```

Product-Specific Information

Integration Server Instance Management

Related Commands

cc delete instances

cc list instances supported products

cc list jobmanager jobs

cc update instances

cc delete instances

Deletes an existing instance of an installed product.

Syntax

- Command Central syntax:

```
cc delete instances  node_alias componentid  [options]
```

- Platform Manager syntax:

```
cc delete instances  componentid  [options]
options:
[--force]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component that you want to delete.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see"Options for the Commands" on page 31.

Usage Notes

- The command returns job information that you can monitor using the cc list job manager command.
- You must stop an Integration Server instance before deleting the instance.

Example When Executing on Command Central

To delete a run-time component with ID "integrationServer-default" that is installed in the installation with alias name "sag01":

```
cc delete instances sag01 integrationServer-default
```

Examples When Executing on Platform Manager

To delete a run-time component with ID "integrationServer-default":

```
cc delete instances integrationServer-default
```

Related Commands

cc create instances

cc list instances supported products

cc list instances supported products

Retrieves a list of products that support instance management.

Syntax

- Command Central syntax:

  ```
  cc list instances node_alias supportedproducts [options]
  ```

- Platform Manager syntax:

  ```
  cc list instances supportedproducts [options]
  ```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the product is installed. You can view a list of installations and their aliases using cc list landscape nodes.

Argument or Option	Description
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see"Options for the Commands" on page 31.

Example When Executing on Command Central

To retrieve a list of the products that support instance management in the installation with alias name “sag01”:

```
cc list instances sag01 supportedproducts
```

Example When Executing on Platform Manager

To retrieve a list of the products that support instance management in the installation:

```
cc list instances supportedproducts
```

Related Commands

cc create instances

cc update instances

cc update instances

Updates configuration properties of an existing instance of an installed product. For example, you might want to update a list of Integration Server packages.

Syntax

- Command Central syntax:

```
cc update instances  node_alias componentid
[key=value] | [-i file{.xml|.json|.properties}]  [options]
```

- Platform Manager syntax:

```
cc update instances  componentid
[key=value] | [-i file{.xml|.json|.properties}]  [options]
```

Arguments and Options

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed.

Argument or Option	Description
	You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required. Specifies the ID of the run-time component for which you want to update configuration properties.
[*key=value*]	Optional. A list of properties that describe the elements of the new instance, such as name and port settings. The properties included in this list are product specific.
[-i [*file*{.xml|.json|.properties}]	Optional. Identifies an input file that contains the new configuration data for the run-time component. For more information, see "--input | -i" on page 42. For the correct format of an XML properties file, see the Properties class in the Oracle Java Platform Standard Edition API specification.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see"Options for the Commands" on page 31.

Example When Executing on Command Central

To update the “WmBusinessRules” package on an Integration Server with ID “integrationServer-default” that is installed in the installation with alias name “sag01”:

```
cc update instances sag01 integrationServer-default
package.list=WmBusinessRules
```

Examples When Executing on Platform Manager

- To update the “WmBusinessRules” package on an Integration Server with ID “integrationServer-default”:

  ```
  cc update instances integrationServer-default
  package.list=WmBusinessRules
  ```

- To update configuration properties for an installed run-time component with ID “OSGI-IS_default” using configuration data in the instance-settings.properties file, located in the current directory:

  ```
  cc update instances OSGI-IS_default -i instance-settings.properties
  ```

- To update configuration properties for an installed run-time component with ID "OSGI-IS_default" using configuration data in the instance-settings.xml file, located in the current directory:

  ```
  cc update instances OSGI-IS_default -i instance-settings.xml
  ```

Related Commands

cc create instances

cc delete instances

cc list instances supported products

8 Jobmanager Jobs Commands

cc list jobmanager jobs

Lists information about long-running jobs. A long-running job is an operation that requires more than a few seconds to complete, for example, the execution of a `cc exec lifecycle` command might take several seconds to complete.

Syntax

- Not supported by Command Central
- Platform Manager syntax:

```
cc list jobmanager jobs  [jobid]  [options]

 options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
`[jobid]`	Optional. Specifies the ID of the job for which you want to retrieve information.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- If you omit `[jobid]` the command retrieves the information for all long-running jobs in the installation that Platform Manager manages.

Examples When Executing on Platform Manager

- To retrieve information for all the long-running jobs in the installation that the Platform Manager server with host name “rubicon2” and port “8092” manages, using the authorization of the user with user name “Administrator” and password “manage”:

```
cc list jobmanager jobs --server http://rubicon2:8092/spm
```

```
--username Administrator --password manage
```

- To retrieve information for the job with ID “5” that is running in the installation that the Platform Manager server with host name “rubicon2” and port “8092” manages, and have the output returned to the console in XML format:

  ```
  cc get jobmanager jobs 3 --server http://rubicon2:8092/spm --format xml
  ```

  Because the `{--username | -u}` and `{--password | -p}` options are not specified, the command uses the default user name and password. For more information, see "--username | -u" on page 52 and "--password | -p" on page 49.

Related Commands

cc get monitoring

cc exec lifecycle

9 Landscape Commands

cc add landscape environments nodes

Adds one or more existing installations (also known as *nodes*) to a specified environment.

Syntax

- Command Central syntax:

```
cc add landscape environments  env_alias
nodes nodeAlias=alias1 [nodeAlias=alias2 ... nodeAlias=aliasn]
[options]

 options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
env_alias	Required. Specifies the alias name of the environment to which you want to add one or more installations.
nodes	Required. Specifies a required keyword indicating you are adding installations (also known at *nodes*) to an environment.
nodeAlias=*alias1* [nodeAlias=*alias2* ... *nodeAlias=aliasn*]	Required. Specifies the alias name(s) of one or more installation(s) that you want to add to the environment.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Use the cc create landscape environments command to create an environment.

- Use the cc create landscape nodes command to create installations that Command Central manages and that you can then add to an environment.
- If you specify installation alias names both on the command line and in an input data file using the `{--input | -i}` option, the command ignores the alias names on the command line and uses only those specified in the input data file.
- You can add the same installation to multiple environments.

Examples When Executing on Command Central

In the following commands the `{--server | -s}`, `{--username | -u}`, and `{--password | -p}` options are not specified. As a result, the command uses the default server, user name, and password. For more information, see "--server | -s" on page 50, "--username | -u" on page 52, and "--password | -p" on page 49.

- To add the installation with alias name “sag01” to the environment with alias name “dev1”:

  ```
  cc add landscape environments dev1 nodes nodeAlias=sag01
  ```

- To add the installations with alias names “is02” and “mws02” to the environment with alias name “env2”:

  ```
  cc add landscape environments env2 nodes nodeAlias=is02 nodeAlias=mws02
  ```

Related Commands

cc create landscape environments

cc get landscape environments

cc list landscape environments

cc remove landscape environments nodes

cc create landscape nodes

cc get landscape nodes

cc list landscape nodes

cc create landscape environments

Creates a new environment that you want to use to manage a collection of installations.

Syntax

- Command Central syntax:
  - To specify the data for the new environment on the command line:

    ```
    cc create landscape environments  alias=env_alias [name=name]
    [description=description]  [options]
    ```

  - To specify the data for the new environment in an input data file:

```
cc create landscape environments
{--input | -i} filename{.xml|.json}  [options]

 options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--media-type | -m} content-type]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
`alias=env_alias`	Required. Specifies the alias name you want to assign to the new environment. The name must be unique among all environments that Command Central manages. Valid characters are ASCII characters, numbers, hyphen (-), underscore (_), and period (.). Spaces are not allowed.
`[name=name]`	Optional. Specifies the display name you want to assign to the environment. If you use a value that includes spaces, place quotes around the value, for example: `name="Dev Environment"` If you do not specify a display name, the command uses the value you supply for the environment alias name.
`[description=description]`	Optional. Specifies a description for the new environment. If you use a value that includes spaces, place quotes around the value, for example: `description="A description with spaces"`
`[{--input | -i} filename{.xml|.json}]`	Optional. Identifies an input file that contains the data for the new environment. For more information, see "--input | -i" on page 42. **Tip:** To determine how to specify the data in the input file, use cc get landscape environments to retrieve data for an existing environment. For example,

Argument or Option	Description
	if you want to use an XML input file, use cc get landscape environments with the `--format xml` option to retrieve the data in XML format.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- After creating the new environment, use cc add landscape environments nodes to add existing installations to the environment.

Example When Executing on Command Central

To create a new environment with the display name “Development1”, the alias name “dev1”, and a description “Environment to test latest release”:

```
cc create landscape environments name=Development1 alias=dev1
description="Environment to test latest release" --password secret
```

Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password

Related Commands

cc add landscape environments nodes

cc delete landscape environments

cc get landscape environments

cc list landscape environments

cc remove landscape environments nodes

cc update landscape environments

cc delete landscape environments

Deletes a specified environment.

Syntax

- Command Central syntax:

  ```
  cc delete landscape environments  [env_alias]  [options]
  ```

```
 options:
[{--debug | -d}]
[{--error | -r} file]
[--force]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
env_alias	Optional. Specifies the alias name of the environment you want to delete.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- If you omit *env_alias*, the command deletes all environments.
- When you delete an environment, the installations in that environment are not deleted. They are still under Command Central management, but no longer assigned to the environment.
- If you want to remove an installation from the environment, use the cc remove landscape environments nodes command.
- If you want to remove an installation from Command Central management, use the cc delete landscape nodes command.

Example When Executing on Command Central

To delete the environment with the alias name “dev1”using the authorization of the user with user name “Administrator” and password “manage”:

```
cc delete landscape environments dev1 --username Administrator
--password manage
```

Because the {--server | -s} option is not specified, the command uses the default server. For more information, see "--server | -s" on page 50.

Related Commands

cc create landscape environments

cc get landscape environments

cc get landscape environments

Retrieves information about a specified environment. Information about an environment can include:

- Alias name
- Display name
- Description, or null if none is assigned
- Information about the installations in the environment

Syntax

- Command Central syntax:

```
cc get landscape environments  env_alias [nodes]  [options]

 options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
env_alias	Required. Specifies the alias name of the environment whose information you want to retrieve.
[nodes]	Optional. Indicates you want the command to return the information about the installations in the environment.

Argument or Option	Description
	If you omit the `nodes` parameter, the returned information will not include the list of installations in the environment.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information for the “dev1” environment, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in XML format:

  ```
  cc get landscape environments dev1 --format xml --server
  http://rubicon:8090/cce --username Administrator --password manage
  ```

- To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information for the “dev1” environment and include information about its installations, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in JavaScript Object Notation format:

  ```
  cc get landscape environments dev1 --format json --server
  http://rubicon:8090/cce --username Administrator --password manage
  ```

Related Commands

cc create landscape environments

cc delete landscape environments

cc list landscape environments

cc update landscape environments

cc list landscape environments

Lists environments in the landscape. Information about an environment can include:

- Alias name
- Display name if one is assigned; otherwise null
- Description if one is assigned; otherwise null
- List of installation aliases that belong to the environment

Syntax

- Command Central syntax:

```
cc list landscape environments  [env_alias]  [options]

 options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
[*env_alias*]	Optional. Specifies the alias name of the environment whose information you want to retrieve. If you do not specify an alias name, the command lists information for all environments.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To list all environments that the Command Central with host name “rubicon” and port “8090” manages, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “environlist” in XML format:

```
cc list landscape environments --format xml --output environlist
--server http://rubicon:8090/cce --username Administrator --password manage
```

Related Commands

cc create landscape environments

cc delete landscape environments

cc get landscape environments

cc remove landscape environments nodes

Removes one or more installations from a specified environment.

Syntax

- Command Central syntax:

```
cc remove landscape environments  env_alias
[nodes nodeAlias=alias1 [nodeAlias=alias2 ... nodeAlias=aliasn]]  [options]

 options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
env_alias	Required. Specifies the alias name of the environment from which you want to remove one or more installations.
nodeAlias=*alias1* [nodeAlias=*alias2* ... *nodeAlias=aliasn*]	Optional. Specifies the alias name(s) of one or more installations that you want to remove from the environment. If you do not specify alias names, the command removes all installations from the specified environment.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The installations that you remove from the environment are not deleted or uninstalled. They are still managed by Command Central, but are no longer associated with the environment.
- If you want to remove an installation from Command Central management, use the cc delete landscape nodes command.

Example When Executing on Command Central

To remove the installations with alias names “mws02” and “is02” from the environment with alias name “env2”:

```
cc remove landscape environments env2 nodeAlias=mws02 nodeAlias=is02
--password secret
```

Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password

Related Commands

cc add landscape environments nodes

cc create landscape environments

cc delete landscape environments

cc get landscape environments

cc list landscape environments

cc update landscape environments

cc delete landscape nodes

cc get landscape nodes

cc list landscape nodes

cc update landscape environments

Updates the display name and/or description assigned to an existing environment.

Syntax

- Command Central syntax:
  - To specify the updated data for the environment on the command line:

    ```
    cc update landscape environments  env_alias  [name=name]
    [description=description]  [options]
    ```

- To specify the updated data for the environment in an input data file:

```
cc update landscape environments  env_alias
{--input | -i} filename{.xml|.json}  [options]

 options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--media-type | -m} content-type]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
env_alias	Required. Specifies the alias name of the environment whose description you want to update.
[name=*name*]	Optional. Specifies the updated display name for the environment. If you use a value that includes spaces, place quotes around the value, for example: `name="Dev Environment"`
[description=*description*]	Optional. Specifies the updated description for the environment. If you use a value that includes spaces, place quotes around the value, for example: `description="A description with spaces"`
[{--input | -i} *filename*{.xml|.json}]	Optional. Identifies an input file that contains the updated data for the environment. For more information, see "--input | -i" on page 42. **Tip:** To determine how to specify the data in the input file, use cc get landscape environments to retrieve data for the environment you want to update. For example, if you want to use an XML input file, use cc get landscape environments with the `--format xml` option to retrieve the data in XML format.

Argument or Option	Description
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- You must specify at least one of the `name` or `description` arguments to indicate the item that you want to update for the environment.

Example When Executing on Command Central

To update the description of an environment with the alias name “dev1” to use the description, “Development version”, use the following command:

```
cc update landscape environments dev1 description="Development version"
--password secret
```

Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Related Commands

cc create landscape environments

cc delete landscape environments

cc get landscape environments

cc list landscape environments

cc create landscape nodes

Adds an installation (also known as a *node*) that you want to manage via Command Central.

Syntax

- Command Central syntax:
  - To specify the data for the new landscape on the command line:

    ```
    cc create landscape nodes  alias=node_alias url=url [name=name]
    [description=description]  [options]
    ```

  - To specify the data for the new landscape in an input data file:

    ```
    cc create landscape nodes  {--input | -i} filename{.xml|.json}
    [options]
    ```

```
 options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--media-type | -m} content-type]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
`alias=node_alias`	Required. Specifies the alias name you want to assign to the installation. The name must be unique among all installations that Command Central manages. Valid characters are ASCII characters, numbers, hyphen (-), underscore (_), and period (.). Spaces are not allowed.
`url=url`	Required. Specifies the URL of the Platform Manager that manages the installation. For example: `http://rubicon:8092` When specifying the URL, if you omit the port number, the command uses “8092”, which is the default port for a Platform Manager server. **Note:** Do not specify a URL that uses the HTTPS protocol. The HTTPS protocol is not supported.
`[name=name]`	Optional. Specifies the display name you want to assign to the installation. If you use a value that includes spaces, place quotes around the value, for example: `name="my installation"` If you do not specify a display name, the command uses the value you specify for the alias name.
`[description=description]`	Optional. Specifies a description for the installation. If you use a value that includes spaces, place quotes around the value, for example:

Argument or Option	Description
	`description="A description with spaces"`
`[{--input | -i} filename{.xml|.json}]`	Optional. Identifies an input file that contains the data for the new node. For more information, see "--input | -i" on page 42. **Tip:** To determine how to specify the data in the input file, use cc get landscape nodes to retrieve data for an existing node. For example, if you want to use an XML input file, use cc get landscape nodes with the `--format xml` option to retrieve the data in XML format.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Use the `cc create landscape nodes` command to create an installation that is not associated with an environment. After creating the installation, you can use the cc add landscape environments nodes command to associate the installation with an environment.

Example When Executing on Command Central

To add an installation managed by the Platform Manager with the URL “http://spm:8092”, and assign it the display name “My webMethods Server” and alias name “mws01”:

```
cc create landscape nodes name="My webMethods Server" alias=mws01
url=http://spm:8092 --password secret
```

Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password

Related Commands

cc delete landscape nodes

cc get landscape nodes

cc list landscape nodes

cc update landscape nodes

cc add landscape environments nodes

cc delete landscape nodes

Removes an installation from being centrally managed via Command Central.

Syntax

- Command Central syntax:

```
cc delete landscape nodes  [{alias | nodeAlias=alias1 [nodeAlias=alias2 ... nodeAlias=aliasn

 options:
[{--debug | -d}]
[{--error | -r} file]
[--force]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
`[{alias | nodeAlias=alias1 [nodeAlias=alias2 ... nodeAlias=aliasn]`	Optional. Specifies the alias name(s) of the installation(s) you want to remove. If you execute the `cc delete landscape nodes` command without supplying alias names, the command removes all installations it is currently managing. To remove a single installation, supply its alias name using the following format: `cc delete landcape nodes alias` To remove multiple installations, supply each name using the `nodeAlias=alias` format to identify each node to remove.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The `cc delete landscape nodes` command does not physically delete the installation(s). It just removes the installation(s) from Command Central management.
- To remove an installation from a specific environment, use the cc remove landscape environments nodes command.

Examples When Executing on Command Central

In the following commands the `{--server | -s}`, `{--username | -u}`, and `{--password | -p}` options are not specified. As a result, the command uses the default server, user name, and password. For more information, see "--server | -s" on page 50, "--username | -u" on page 52, and "--password | -p" on page 49.

- To remove the installation with alias “mws01”:

  ```
  cc delete landscape nodes mws01
  ```

- To remove the installations with alias names “mws01” and “sag01”:

  ```
  cc delete landscape nodes nodeAlias=mws01 nodeAlias=sag01
  ```

- To remove all installations:

  ```
  cc delete landscape nodes
  ```

Related Commands

cc create landscape nodes

cc get landscape nodes

cc list landscape nodes

cc update landscape nodes

cc remove landscape environments nodes

cc exec landscape nodes generateNodeId

Generates or regenerates a unique ID for an existing installation.

Note: The installation ID is not the same as the alias name for an installation.

Syntax

- Command Central syntax:

  ```
  cc exec landscape nodes  node_alias generateNodeId  [options]

   options:
  [{--debug | -d}]
  [{--error | -r} file]
  [{--log | -l} file]
  ```

```
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
node_alias	Required. Specifies the alias name of the installation for which you want to generate an ID. You can view a list of installations and their aliases using cc list landscape nodes.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Typically, you should not need to generate or regenerate an ID for an installation. Command Central generates an ID for an installation when it is originally added to Command Central management, for example, by executing the cc create landscape nodes command.

  You might regenerate an ID if you have an installation with a duplicate ID. This can occur, for example, if you copy an image of an installation.

- The `cc exec landscape nodes` command stores the newly generated ID in its proper location. The command does not return the new ID as output.

Example When Executing on Command Central

To generate an ID for the installation with alias name “sag01” using the authorization of the user with user name “Administrator” and password “manage”:

```
cc exec landscape nodes sag01 generateNodeId --username Administrator
--password manage
```

Because the `{--server | -s}` option is not specified, the command uses the default server. For more information, see "--server | -s" on page 50.

Related Commands

cc create landscape nodes

cc delete landscape nodes

cc get landscape nodes

Retrieves information about a specified installation. Information about an installation can include:

- Alias name
- Display name
- Description, or null if none is assigned
- URL of the Platform Manager that manages the installation
- Status of the Platform Manager that manages the installation

Syntax

- Command Central syntax:

```
cc get landscape nodes  alias  [options]

 options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
node_alias	Required. Specifies the alias name of the installation for which you want to retrieve information. You can view a list of installations and their aliases using cc list landscape nodes.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For

Argument or Option	Description
	a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The information for an installation can include the status of the Platform Manager that manages the installation. The status is:
  - ONLINE when Command Central can connect with the Platform Manager.
  - OFFLINE when Command Central cannot connect to the Platform Manager, for example, if Platform Manager is not running or if there are other connection issues.
- If a Platform Manager is OFFLINE, the command only retrieve the Platform Manager manager status for the installation because the command relies on the Platform Manager to provide the other installation information it retrieves.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve information for the installation with alias name “sag01”using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the console in JavaScript Object Notation format:

```
cc get landscape nodes sag01 --format json --server http://rubicon:8090/cce
--username Administrator --password manage
```

Related Commands

cc create landscape nodes

cc delete landscape nodes

cc list landscape nodes

cc update landscape nodes

cc list landscape nodes

Lists the installations that Command Central manages. Information about an installation can include:

- Alias name
- Display name
- Description, or null if none is assigned
- URL of the Platform Manager that manages the installation
- Status of the Platform Manager that manages the installation

Syntax

- Command Central syntax:

```
cc list landscape nodes  [node_alias]  [options]

 options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
[*node_alias*]	Optional. Specifies the alias name of the installation for which you want to list information.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- If you do not specify the alias of a specific installation, the command lists installations that Command Central manages.
- The information for an installation can include the status of the Platform Manager that manages the installation. The status is:
  - ONLINE when Command Central can connect with the Platform Manager
  - OFFLINE when Command Central cannot connect to the Platform Manager, for example, if Platform Manager is not running or if there are other connection issues
- If a Platform Manager is OFFLINE, the command only retrieve the Platform Manager manager status for an installation because the command relies on the Platform Manager to provide the other installation information it retrieves.

Example When Executing on Command Central

To list all installation that the Command Central with host name “rubicon” and port “8090” manages, using the authorization of the user with user name “Administrator” and password “manage”, and have the information returned to the output file “nodelist” in XML format:

```
cc list landscape nodes --format xml --output nodelist
--server http://rubicon:8090/cce --username Administrator
--password manage
```

Related Commands

cc create landscape nodes

cc delete landscape nodes

cc get landscape nodes

cc update landscape nodes

cc update landscape nodes

Updates the properties assigned to an installation, for example, the display name or description.

Syntax

- Command Central syntax:
  - To specify the updated data for the landscape on the command line:

    ```
    cc update landscape nodes  node_alias [name=name]
    [description=description]  [options]
    ```

  - To specify the updated data for the landscape in an input data file:

    ```
    cc create landscape nodes  node_alias
    {--input | -i} filename{.xml|.json}  [options]
    ```

  ```
  options:
  [{--debug | -d}]
  [{--error | -r} file]
  [{--log | -l} file]
  [{--media-type | -m} content-type]
  [{--password | -p} password]
  [{--quiet | -q}]
  [{--server | -s} url]
  [{--username | -u} user_name]
  ```

- Not applicable to Platform Manager

Arguments and Options

Argument or Option	Description
node_alias	Required. Specifies the alias name of the installation whose description you want to update. You can view a list of installations and their aliases using cc list landscape nodes.
[name=*name*]	Optional. Specifies the updated display name for the installation. If you use a value that includes spaces, place quotes around the value, for example: `name="My installation"`
[description=*description*]	Optional. Specifies the updated description for the installation. If you use a value that includes spaces, place quotes around the value, for example: `description="A description with spaces"`
[{--input | -i} *filename*{.xml|.json}]	Optional. Identifies an input file that contains the updated data for the landscape. For more information, see "--input | -i" on page 42. **Tip:** To determine how to specify the data in the input file, use the cc get landscape nodes to retrieve data for the node you want to update. For example, if you want to use an XML input file, use cc get landscape nodes with the `--format xml` option to retrieve the data in XML format.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- You must specify at least one of the `name` or `description` arguments to indicate the item that you want to update for the installation.

Example When Executing on Command Central

To update the installation with alias name “sag01” to use the description, “updated version”:

```
cc update landscape nodes sag01 description="updated version"
--password "secret"
```

Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Related Commands

cc create landscape nodes

cc delete landscape nodes

cc get landscape nodes

cc list landscape nodes

10 License Reports Commands

cc create license-tools reports snapshot

Creates a license compliance snapshot report based on the currently registered nodes in a Command Central instance.

Syntax

- Command Central syntax:

  ```
  cc create license-tools reports snapshot [options]
  ```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

To check the status of the job scheduled to create the license report, use the following command:

```
cc get jobmanager jobs [jobid] --expected-values DONE --wait [seconds]
```

cc delete license-tools reports snapshot

Deletes all generated license reports from Command Central.

Syntax

- Command Central syntax:

  ```
  cc delete license-tools reports snapshot [options]

   options:
  [--force]
  ```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Related Commands

cc delete license-tools reports snapshot reportid

cc get license-tools reports snapshot

cc get license-tools reports snapshot reportid

cc delete license-tools reports snapshot reportid

Deletes an existing license report with the specified unique report identifier.

Syntax

- Command Central syntax:

```
cc delete license-tools reports snapshot reportid [options]

 options:
[--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
`reportid`	Required. The ID of the license report to delete.
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To delete a report with report ID `c4eff6`:

```
cc delete license-tools reports snapshot c4eff6
```

Related Commands

cc delete license-tools reports snapshot

cc get license-tools reports snapshot

cc get license-tools reports snapshot reportid

cc get license-tools reports snapshot

Lists all license reports available on the Command Central server.

Syntax

- Command Central syntax:

  ```
  cc get license-tools reports snapshot [options]
  ```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- Command Central and Platform Manager support the following formats for the output report data:
  - comma-separated values
  - tab-separated values
  - XML
  - JSON
- The output data for each report in the list includes the report ID, the name of the user who created the report, the date the report was created, and the compliance status of the report.
- When an existing report fails checksum verification, the report is not included in the output list.

Related Commands

cc create license-tools reports snapshot

cc delete license-tools reports snapshot

cc delete license-tools reports snapshot reportid

cc get license-tools reports snapshot reportid

cc get license-tools reports snapshot reportid

Obtains information about a license report with the specified unique report identifier.

Syntax

- Command Central syntax:

  ```
  cc get license-tools reports snapshot reportid [options]
  ```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
reportid	Required. The ID of the report for which to obtain information.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

To obtain information about a report with report ID `c4eff6`:

```
cc get license-tools reports snapshot c4eff6
```

Related Commands

cc create license-tools reports snapshot

cc delete license-tools reports snapshot

cc delete license-tools reports snapshot reportid

cc get license-tools reports snapshot

cc get license-tools reports snapshot output PDF

Generates a PDF file for an existing license report.

Syntax

- Command Central syntax:

  ```
  cc get license-tools reports snapshot reportid --output-format pdf
  --output filename.pdf
  ```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
reportid	Required. The ID of the license report for which to generate a PDF file.
`--output-format pdf`	Required. Specifies that the output is generated in PDF format.
`--output-`*filename.pdf*	Required. Specifies a name for the output PDF file. For more information, see "--output	-o" on page 47.

Usage Notes

Platform Manager sends a request to the Command Central server, with the Accept HTTP header value set to `application/pdf` to indicate that the license report output data should be generated in PDF format.

Example When Executing on Command Central

To generate a PDF file, named report.pdf, for a license report with ID `c4eff6` and save the generated report file in the current directory:

```
cc get license-tools reports snapshot c4eff6 --output-format pdf
--output report.pdf
```

Related Commands

cc create license-tools reports snapshot

cc get license-tools reports snapshot output XML

cc get license-tools reports snapshot output XML

Generates an existing license report in XML format.

Syntax

- Command Central syntax:

  ```
  cc get license-tools reports snapshot reportid --output-format xml
  --output filename.xml
  ```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
reportid	Required. The ID of the license report for which to generate an XML file.
`--output-format xml`	Required. Specifies that the output is generated in XML format.
`--output-`*filename.xml*	Required. Specifies a name for the output XML file. For more information, see "--output	-o" on page 47.

Example When Executing on Command Central

To generate an XML file, named report.xml, for a license report with ID `c4eff6` and save the generated report file in the current directory:

```
cc get license-tools reports snapshot c4eff6 --output-format xml
--output report.xml
```

Related Commands

cc create license-tools reports snapshot

cc get license-tools reports snapshot output PDF

11 Lifecycle Commands

cc exec lifecycle

Executes an action against run-time components. You can execute actions to start, stop, pause, and/or resume run-time components.

Syntax

- Command Central syntax:
  - To execute an action against a specified component:

    ```
    cc exec lifecycle action node_alias componentid [options]
    ```

  - To execute an action against run-time components that meet specified search criteria:

    ```
    cc exec lifecycle action [criteria] [options]
    ```

- Platform Manager syntax:

  ```
  cc exec lifecycle components componentid action [options]
  ```

```
 options:
[{--accept | -a} content_type]
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

Arguments and Options

Argument or Option	Description
action	Required. Specifies the action you want to take against the run-time component. Supply one of the following actions: - `start` - starts the run-time component - `startindebugmode` - starts the run-time component in debug mode - `startinsafemode` - starts the run-time component in safe mode - `stop` - stops the run-time component - `restart` - stops, then restarts the run-time component

Argument or Option	Description
	■ `pause` - pauses the run-time component ■ `resume` - resumes previously paused run-time component Run-time components might support all or just a subset of the actions. For information about the supported actions for a run-time component, see information in this reference for the product with which the run-time component is associated.
node_alias	Command Central only. Required when you do not specify search criteria. Specifies the alias name of an installation. You can determine installation alias names using the cc list landscape nodes command.
componentid	Required. Specifies the component ID of a run-time component on which to act. You can determine the IDs for run-time components using the cc list inventory components command. **Note:** Command Central only requires the *componentid* when you do not specify search criteria. When executing the command against a Platform Manager, specify the *componentid**before*** the action in the command syntax.
[*criteria*]	Command Central only. Optional. Specifies to act only on the run-time components that match the search criteria you specify. For more information, see "Specifying Search Criteria for Lifecycle Commands" on page 165.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The command returns job information, that includes information such as the job ID and job status.

- You can execute the `cc get monitoring runtimestatus` command after executing the `cc exec lifecycle` command to determine when the requested action is complete. Use the `{--expected-values |-e}`, `{--check every |-c}`, and `{--wait |-w}` options with the `cc get monitoring runtimestatus` command to specify the results for which to check and how often to check for the results. For more information, see "cc get monitoring" on page 168.

Examples When Executing on Command Central

- To start the run-time component on the installation with alias name “sag01” and component ID “OSGI-SPM”:

  ```
  cc exec lifecycle start sag01 OSGI-SPM
  ```

  Because the `{--server | -s}`, `{--username | -u}`, and `{--password | -p}` options are not specified, the command uses the default server, user name, and password. For more information, see "--server | -s" on page 50, "--username | -u" on page 52, and "--password | -p" on page 49.

- To stop all run-time components that contain “OSGI” in the component display name:

  ```
  cc exec lifecycle stop displayName=*OSGI* --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Examples When Executing on Platform Manager

- To restart the run-time component with ID “OSGI-SPM” that is installed in the installation managed by the Platform Manager with host name “rubicon2” and port “8092”, using the authorization of the user with user name “Administrator” and password “manage”:

  ```
  cc exec lifecycle components OSGI-SPM restart --server
  http://rubicon2:8092/spm --username Administrator --password manage
  ```

- To stop the run-time component with ID “OSGI-IS” that is installed in the installation managed by the Platform Manager with host name “rubicon2” and port “8092”:

  ```
  cc exec lifecycle components OSGI-IS stop
  --server http://rubicon2:8092/spm --password secret
  ```

  Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Product-Specific Information

Lifecycle Actions for Broker Server

Related Commands

cc list landscape nodes

cc list inventory components

cc list jobmanager jobs

Specifying Search Criteria for Lifecycle Commands

When executing the `cc exec lifecycle` command on a Command Central server, you can specify search criteria to identify the run-time components against the command should act. Supply the search criteria using the following format:

`[logicalOperator=OR]` *`property1=value1 property2=value2 ...`*

For the search criteria, you specify property values to match, for example `runtimeComponentId=OSGI-CCE`, where `runtimeComponentId` is the property and the value to match is `OSGI-CCE`.

Specifying the Value

When specifying the value, you can include the `*` pattern-matching character to match multiple characters. For example, if you want to act only on run-time components with "cce" anywhere in their display names, use the following search criterion:

```
displayName=*cce*
```

Important: The search is case-sensitive.

Property Names You Can Use in the Search Criteria

- `nodeName`
- `nodeAlias`
- `nodeUrl`
- `environmentName`
- `environmentAlias`
- `runtimeComponentInfoId`
- `runtimeComponentId`
- `runtimeComponentDisplayName`
- `runtimeComponentProductId`
- `runtimeComponentCategory`
- `runtimeComponentRuntimeStatus`
- `runtimeComponentRuntimeParentId`

Logical Operators Used When Specifying Multiple Search Properties

If you specify multiple search items, by default, the command performs an AND operation to return results that match all the specified criteria. For example, to act only

on run-time components with “cce” anywhere in their display names *and* that are part of products that have the ID “OSGI”, use the following search criteria:

```
displayName=*cce* productId=OSGI
```

You can use an OR operation with two properties. To do so, specify the `logicalOperator=OR` argument. For example, to act on run-time components with “cce” anywhere in their display names *or* that are part of products that have the ID “OSGI”, use the following search criteria:

```
displayName=*cce* logicalOperator=OR productId=OSGI
```

Related Commands

cc exec lifecycle

cc list inventory components

12 Monitoring Commands

cc get monitoring

Retrieves the run-time statuses, run-time states, or states of run-time components.

- Run-time status indicates whether a run-time component is running or not. Examples of a run-time status are ONLINE or STOPPED.
- Run-time state indicates the health of a run-time component by providing (key performance indicators (KPIs) for the component. Each KPI provides information about the current use, marginal use, critical use, and maximum use. For example, a component might display a KPI for the amount of memory that would include the current memory use, when memory use is considered marginal, when memory use is considered critical, and the maximum memory use allowed.
- State provides both the run-time status and the run-time state.

For a list and description of run-time statuses and run-time states for a specific run-time component, see information in this reference for the product with which the run-time component is associated.

Syntax

- Command Central syntax:

```
cc get monitoring  {runtimestatus | runtimestate | state} node_alias
 componentid  [options]
```

- Platform Manager syntax:

```
cc get monitoring  {runtimestatus | runtimestate | state} componentid
[options]
```

```
 options:
[{--check-every | -c} seconds]
[{--debug | -d}]
[{--error | -r} file]
[{--expected-values | -e} values]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
[{--wait | -w} seconds]
```

Arguments and Options

Argument or Option	Description
{runtimestatus | runtimestate | state}	Required. Specifies whether you want to retrieve run-time statuses, run-time states, or state.

Argument or Option	Description
node_alias	Command Central only. Required. Specifies the alias name of the installation on which the run-time component is installed. You can view a list of installations and their aliases using cc list landscape nodes.
componentid	Required for `runtimestatus` and `rumtimestate`. Not applicable for `state`. Specifies the ID of the run-time component for which you want to retrieve information. You can determine the IDs for run-time components using cc list inventory components.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The following are general descriptions of run-time statuses:
  - FAILED when a run-time component failed, for example, ended unexpectedly.
  - ONLINE when a run-time component is running.
  - PAUSED when a run-time component is paused.
  - STARTING when a run-time component is starting.
  - STOPPED when a run-time component is not running.
  - STOPPING when a run-time component is stopping.
  - UNKNOWN when the status cannot be determined.
  - UNRESPONSIVE when a run-time component is running, but is unresponsive.

  Note: A specific run-time component might support only a subset of the statuses.

Examples When Executing on Command Central

- To retrieve the run-time status of the run-time component that has the ID “OSGI-SPM” and is installed in the installation with alias name “sag01” and have the output returned to the console in XML format:

  ```
  cc get monitoring runtimestatus sag01 OSGI-SPM --format xml
  --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server

| -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

- To initiate the shutdown of the Integration Server with the component ID “OSGI-IS” running in the installation “sag01”, then execute the `cc get monitoring runtimestatus` command to wait 60 seconds for the command to complete and return the expected results “STOPPED”, checking for results every 5 seconds:

  ```
  cc exec lifecycle stop sag01 OSGI-IS --password secret
  cc get monitoring runtimestatus sag01 OSGI-IS --expected-values STOPPED
  --wait 60 --check-every 5 --password secret
  ```

  Because the `{--server | -s}` and `{--username | -u}` options are not specified, the command uses the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Example When Executing on Platform Manager

To retrieve the state of the run-time component that has the ID “OSGI-SPM” and is installed in the installation that the Platform Manager server with host name “rubicon2” and port “8092”” manages, and have the output returned to the console in XML format:

```
cc get monitoring state OSGI-SPM --format xml
--server http://rubicon2:8092/spm --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Product-Specific Information

Run-time Monitoring Statuses for IntegrationServer-instanceName

Run-time Monitoring States for IntegrationServer-instanceName

Monitoring Run-time Statuses for webMethods Broker

Monitoring Run-time States for webMethods Broker

Related Commands

cc list inventory components

cc list monitoring alerts

Lists the alerts for a specified run-time component.

Syntax

- Command Central syntax:

  ```
  cc list monitoring alerts  [node_alias [componentid]]  [options]

   options:
  [{--debug | -d}]
  ```

```
[{--error | -r} file]
[{--format | -f} {tsv args | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not supported by Platform Manager

Arguments and Options

Argument or Option	Description
[*node_alias*] [*componentid*]]	Optional. Specifies the alias name of the installation and the ID of the run-time component for which you want to retrieve information. To specify *componentid*, you must specify *node_alias*. You can view a list of installations and their aliases using cc list landscape nodes. You can determine the IDs for run-time components using cc list inventory components.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to list the alerts for the run-time component that has the ID “OSGI-SPM” and is installed in the installation with alias name “sag01”, and have the output returned to the console in XML format:

```
cc list monitoring alerts sag01 OSGI-SPM --format xml
--server http://rubicon:8090/cce --password secret
```

Because the {--username | -u} option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

13 Repositories Commands

cc add repository fixes

Registers a fix repository.

Syntax

- Command Central syntax:

  ```
  cc add repository fixes  node_alias path=path  [options]
  ```

- Platform Manager syntax:

  ```
  cc add repository fixes path=path  [options]
  ```

Arguments and Options

Argument or Option	Description
node_alias	Required. Only for Command Central. Specifies the alias name of the Platform Manager to which you want to add the fix repository.

path=*path*	Required. The path to the fix repository to register.
	■ If the fix repository is remote, specify the URI with http or https.
	■ If the fix repository is local, the path may refer either to an image archive created with Software AG Update Manager (containing images or products), or to a local directory. When the location is a directory:
	  ■ If the directory that does not exist or is empty, Platform Manager creates the directory and populates it with and empty p2 repository.
	  ■ If the directory exists and is not empty, ensure that it is a valid p2 repository of products.

[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

After adding a fix repository by using the discover or register command, Platform Manager creates a new configuration instance that you use to configure the repository. For example, you can add access credentials to a remote repository by using the new configuration instance for a registered fix repository as follows:

1. Execute the following command:

   ```
   cc get configuration data node-spm OSGI-SPM-ENGINE
   REPOSITORY-empower-fix-image.zip
   ```

2. Save the result to a file with name “repoCredentials.prop”.

3. In a text editor, edit the repoCredentials.prop file as follows:

   ```
   user=EmpowerUserEmailAddress
   password=EmpowerUserPassword
   ```

4. Execute the following commands:

   ```
   cc update configuration data node-spm
   OSGI-SPM-ENGINE REPOSITORY-empower-fix-image.zip
   -i repoCredentials.prop
   cc get configuration data node-spm
   OSGI-SPM-ENGINE REPOSITORY-empower-fix-image.zip
   ```

   The result contains the updated password and username.

Example When Executing on Command Central

To add an image file with name “image.zip” as a repository in the Platform Manager registered as “is-dev”:

```
cc add repository fixes is-dev path=file:///home/neta/work/image.zip
```

If you do not specify a value for CC_SERVER, the request is sent to the Command Central server on “localhost:8090”.

Example When Executing on Platform Manager

To add an image file with name “image.zip” as a repository in the Platform Manager hosted on a server with host name “rubicon” and port “8092”:

```
cc add repository fixes path=file:///home/neta/work/image.zip
--server http://rubicon:8092/spm
```

Related Commands

cc exec repository discover

cc exec repository register

cc list repository

cc list repository discover

cc add repository products

Registers product repositories.

Syntax

- Command Central syntax:

  ```
  cc add repository products  node_alias path=path [options]
  ```

- Platform Manager syntax:

  ```
  cc add repository products  path=path [options]
  ```

Arguments and Options

Argument or Option	Description
node_alias	Required. Only for Command Central. Specifies the alias name of the Platform Manager to which you want to add the product repository.

path=*path*	Required. The path to the product repository to register.
	■ If the product repository is remote, specify the URI with http or https.
	■ If the product repository is local, the path may refer either to an image archive created with Software AG Installer (containing images or products), or to a local directory. When the location is a directory:
	  ■ If the directory that does not exist or is empty, Platform Manager creates the directory and populates it with and empty p2 repository.
	  ■ If the directory exists and is not empty, ensure that it is a valid p2 repository of products.

[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

After adding a product repository by using the discover or register command, Platform Manager creates a new configuration instance that you use to configure the repository. For example, you can add access credentials to a remote repository by using the new configuration instance for a registered product repository as follows:

1. To select the registered product repository for which you want to configure credentials:
   - Execute the command:

     ```
     cc list repository products node-spm
     ```
   - Select the name of the repository from the list.
2. Execute the following command:

   ```
   cc get configuration data node-spm
   OSGI-SPM-ENGINE REPOSITORY-sdc.softwareag.com-cgi
   -bin-dataservewebM95.cgi-image.zip
   ```
3. Save the result to a file with name “repoCredentials.prop”.
4. In a text editor, edit the repoCredentials.prop file as follows:

   ```
   user=EmpowerUserEmailAddress
   password=EmpowerUserPassword
   ```
5. Execute the following commands:

   ```
   cc update configuration data node-spm OSGI-SPM-ENGINE
   REPOSITORY-sdc.softwareag.com-cgi-bin-dataservewebM95.cgi
   -image.zip -i repoCredentials.prop
   cc get configuration data node-spm OSGI-SPM-ENGINE
   REPOSITORY-sdc.softwareag.com-cgi-bin-dataservewebM95.cgi-image.zip
   ```

   The result contains the updated password and username.

Example When Executing on Command Central

To add an existing image file with name “image.zip” (created using the Software AG Installer) as a repository in the Platform Manager registered as “is-dev”:

```
cc add repository products is-dev path=file:///home/neta/work/image.zip
```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on “localhost:8090”.

Example When Executing on Platform Manager

To add an existing image file with name “image.zip” (created using the Software AG Installer) as a repository in the Platform Manager hosted on a server with host name “rubicon” and port “8092”:

```
cc add repository products
path=file:///home/neta/work/image.zip -server http://rubicon:8092/spm
```

Related Commands

cc exec repository discover

cc exec repository products register

cc exec repository register

cc list repository

cc list repository discover

cc create repository

Creates a product or fix repository.

Syntax

- Command Central syntax:

  ```
  cc create repository {products | fixes} name=repo_name [location=URL]
  [options]
  ```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
`name=repo_name`	Required. The name of the product or fix repository to create.
`location=URL`	Optional. A valid URL that points to the location where the repository is created. If `location` points to an installation image file, the image file must exist on the Command Central server. If the image file does not exist, the repository is not created. **Important:** Two repositories cannot point to the same location.
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- To create a repository with name “test” at the “http://test/repo” location:

```
cc create repository products name=test location=http://test/repo
cc create repository fixes name=test location=http://test/repo
```

- To upload an image file named “image.zip” from the current directory to Command Central and create a repository with name “test” that points to that image:

```
cc create repository products name=test -i image.zip
cc create repository fixes name=test -i image.zip
```

Related Commands

cc exec repository register

cc exec repository discover

cc list repository

cc update repository

cc delete repository

Deletes a registered product or fix repository.

Syntax

- Command Central syntax:

```
cc delete repository {products | fixes} repo_name
[deleteImage={true | false}] [options]

 options:
[--force]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
repo name	Required. Specifies the name of the repository that you want to delete.
deleteImage={true | false}	Optional. Whether to delete the image file referenced by the repository from the file system. Valid values: ■ `true` Delete the image file. ■ `false` (default) Do not delete the image file.

Argument or Option	Description
	When you do not include this argument, Command Central does not delete the image file from the file system.
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- To delete a repository with name “repo1” including the image file that the repository references:

```
cc delete repository products REPOSITORY-repo1 deleteImage=true
cc delete repository fixes REPOSITORY-repo1 deleteImage=true
```

- To delete a product repository with name “test” without deleting the image file that the repository refers to:

```
cc delete repository products test
```

Related Commands

cc create repository

cc exec repository discover

cc list repository

cc update repository

cc delete repository with *node_alias*

Deletes a registered product or fix repository on a specified Platform Manager node. The command removes access to the repository without deleting actual repository data.

Syntax

- Command Central syntax:

```
cc delete repository {products | fixes} node_alias repo_name [options]

 options:
[--force]
```

- Platform Manager syntax:

```
cc delete repository {products | fixes} node_alias repo_name [options]
```

Arguments and Options

Argument or Option	Description
node_alias	Required. Only for Command Central. Specifies the alias name of the Platform Manager from which you want to delete the repository.
repo_name	Required. Specifies the name of the repository that you want to delete.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

- To delete a repository with name “repo1” from the Platform Manager registered as “is-dev”:

```
cc delete repository products is-dev REPOSITORY-repo1
cc delete repository fixes is-dev REPOSITORY-repo1
```

Related Commands

cc create repository

cc exec repository discover

cc list repository

cc update repository

cc delete repositories

Deletes all registered product or fix repositories.

Syntax

- Command Central syntax:

```
cc delete repository {products | fixes}[options]

 options:
[--force]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

To prevent loss of information, you are prompted to confirm if you want to delete all repositories of the specified type. Use the `--force` command option to override the confirmation request. For more information, see "--force" on page 39.

Examples When Executing on Command Central

- To delete all registered product repositories after user confirmation:

```
cc delete repository products
```

- To delete all registered fix repositories without user confirmation:

```
cc delete repository products --force
```

Related Commands

cc delete repository

cc delete repository with node_alias

cc exec repository discover

Finds product and fix repositories for the specified host, name, and port and adds the discovered repositories to Command Central.

Syntax

- Command Central syntax:

```
cc exec repository {products | fixes} discover [host=host]
[name=repo_name] [port=port][options]
```

- Platform Manager syntax:

```
cc exec repository discover [host=host] name=repo_name
[port=port][options]
```

Arguments and Options

Argument or Option	Description
`[host=host]`	Optional. The host name or IP address of the system hosting the repositories. If you do not specify a value, Command Central goes to the Empower website.
`[name=repo_name]`	Optional. The name of the repository on the specified host machine. If you do not specify a value, Command Central lists all repositories on the host machine.
`[port=port]`	Optional. The port number of the specified host machine.
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central and Platform Manager

To add a product repository with name “SuiteProd” from a server with host name “sag”:

```
cc exec repository products discover host=sag name=SuiteProd
cc exec repository fixes discover host=sag name=SuiteProd
```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on “localhost:8090”.

Related Commands

cc create repository

cc exec repository products register

cc list repository

cc list repository discover

cc update repository

cc exec repository discover with *node_alias*

Discovers public product or fix repositories.

Syntax

- Command Central syntax:

```
cc exec repository {products | fixes} node_alias discover
[host=host] [name=repo_name] [port=port] [options]
```

- Platform Manager syntax:

```
cc exec repository {products | fixes} discover [host=host]
[name=repo_name]  [port=port] [options]
```

Arguments and Options

Argument or Option	Description
node_alias	Required. Only for Command Central. Specifies the alias name of the Platform Manager for which you want discover repositories.
[host=*host*]	Optional. The host name or IP address of the system hosting the repositories. If you do not specify a value, Command Central and Platform Manager go to the Empower website.
[name=*repo_name*]	Required when you specify a value for `host`. The name of the repository on the specified host machine. If you do not specify a value, Command Central lists all repositories on the host machine.
[port=*port*]	Optional. The port number of the specified host machine.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

After adding a repository by using the discover or register command, Platform Manager creates a new configuration instance that you use to configure the repository. For an example how to add access credentials to a remote repository, see the usage notes section in:

- For product repositories, "cc add repository products" on page 176
- For fix repositories, "cc add repository fixes" on page 174

Example When Executing on Command Central

To add a product repository with name “SuiteProd” from a server with host name “sag” to the Platform Manager registered as “is-dev”:

```
cc exec repository products is-dev discover host=sag name=SuiteProd
```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on “localhost:8090”.

Example When Executing on Platform Manager

To add a fix repository with name “QARepo” from a server with host name “sag” to the Platform Manager hosted on a server with host name “rubicon” and port “8092”:

```
cc exec repository fixes discover host=sag name=QARepo -s rubicon:8092/spm
```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on “localhost:8090”.

Related Commands

cc add repository products

cc list repository

cc list repository discover

cc exec repository products register

Registers a product repository on a specified Platform Manager node and deletes all previously added repositories on the node.

Syntax

- Command Central syntax:

  ```
  cc exec repository products register node_alias repo_name
  [options]
  ```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
node_alias	Required. Specifies the alias name of the Platform Manager on which you want to register the repository.
repo_name	Required. Specified the name of the repository that you want to register.

Argument or Option	Description
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To register a product repository with name “test” on the local Platform Manager:

```
cc exec repository products register local test
```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on “localhost:8090”.

Related Commands

cc create repository

cc delete repository

cc list repository

cc update repository

cc exec repository register

Copies a product or fix repository, including its image file, to a new Platform Manager node.

Syntax

- Command Central syntax:

  ```
  cc exec repository {products | fixes} register node_alias repo_name
  [options]
  ```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
node_alias	Required. Specifies the alias name of the Platform Manager to which you want to copy the repository.
repo_name	Required. Specified the name of the repository that you want to copy.

Argument or Option	Description
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

To copy the repository with name “repo1” to the Platform Manager with name “node1”:

```
cc exec repository products register node1 repo1
cc exec repository fixes register node1 repo1
```

Related Commands

cc create repository

cc delete repository

cc exec repository discover

cc list repository

cc update repository

cc list repository

Lists registered product or fix repositories.

Syntax

- Command Central syntax:

  ```
  cc list repository {products | fixes} [options]
  ```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Examples When Executing on Command Central

To list the registered repositories on a Command Central server with host name "rubicon" and port "8490":

```
cc list repository products -server http://rubicon:8490/cce
cc list repository fixes -server http://rubicon:8490/cce
```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on "localhost:8090".

Related Commands

cc create repository

cc exec repository products register

cc list repository discover

cc update repository

cc list repository with *node_alias*

Lists registered product and fix repositories on a specified Platform Manager node.

Syntax

- Command Central syntax:

  ```
  cc list repository {products | fixes} node_alias [options]
  ```

- Platform Manager syntax:

  ```
  cc list repository {products | fixes} [options]
  ```

Arguments and Options

Argument or Option	Description
node_alias	Required. Only for Command Central. Specifies the alias name of the Platform Manager for which you want to list repositories.
`refresh`	Optional. Whether to send a new request to the specified Platform Manager to update the Command Central server cache with data about the registered repositories. - `true` Sends a request. - `false` (default) Does not send a request.

Argument or Option	Description
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

- To list the registered fix repositories for a Platform Manager registered as “is-dev”:

  ```
  cc list repository fixes is-dev
  ```

- To list the registered product repositories for a Platform Manager registered as “is-dev” on a Command Central server with host name “rubicon” and port “8490”:

  ```
  cc list repository products is-dev -server http://rubicon:8490/cce
  ```

- To update the Command Central server cache and list all registered fix repositories for a Platform Manager registered as “is-dev”:

  ```
  cc list repository fixes is-dev refresh=true
  ```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on “localhost:8090”.

Example When Executing on Platform Manager

To list the registered product repositories for a Platform Manager hosted on a server with host name “rubicon” and port “8092”:

```
cc list repository products -server http://rubicon:8092/spm
```

Related Commands

cc list repository

cc list repository discover

cc list repository discover

Finds product or fix repositories for the specified host, name, and port, but does not add the discovered repositories to Command Central.

Syntax

- Command Central syntax:

  ```
  cc list repository discover [host=host] [name=repo_name]
  [port=port] [options]
  ```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
`[host=host]`	Optional. The host name or IP address of the system hosting the repositories. If you do not specify a value, Command Central goes to the Empower website.
`[name=repo_name]`	Optional. The name of the repository on the specified host machine. If you do not specify a value, Command Central lists all repositories on the host machine.
`[port=port]`	Optional. The port number of the specified host machine.
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To find a product repository with name “SuiteProd” on a server with host name “sag”:

```
cc list repository discover host=sag name=SuiteProd
```

If you do not specify a value for `CC_SERVER`, the request is sent to the Command Central server on “localhost:8090”.

Related Commands

cc create repository

cc exec repository discover

cc exec repository products register

cc list repository

cc update repository

cc update repository

Updates a repository using data from an XML file. The XML file must contain a valid XML representation of the repository and the name of the repository.

Syntax

- Command Central syntax:

```
cc update repository {products | fixes} -i filename.xml [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
`-i filename.xml`	Required. Specified the name of the XML file that contains the repository data. For more information, see "--input	-i" on page 42.
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To update a repository using data from the “repository.xml” file, located in the current directory:

```
cc update repository products  -i repository.xml
cc update repository fixes  -i repository.xml
```

Related Commands

cc create repository

cc exec repository discover

cc exec repository products register

cc list repository

cc update repository details

cc update repository details

Updates a repository description and location.

Syntax

- Command Central syntax:

```
cc update repository {products | fixes} repo_name
[description="description"] [location=path] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
repo_name	Required. Specifies the name of the repository that you want to update.
[description="*description*"]	Optional. The new description of the repository.
[location=*path*]	Optional. The path to the new location of the image file referenced by the repository.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To change the location of the file “image.zip”referenced by the repository with name “repo1”:

```
cc update repository products repo1 location=file:///vmtest/image.zip
cc update repository fixes repo1 location=file:///vmtest/image.zip
```

Related Commands

cc create repository

cc exec repository discover

cc exec repository products register

cc list repository

cc update repository

14 Resources Commands

cc list resources icons

Lists information about the installed icons. This command is useful if you want to use the cc update inventory components command to update the icon associated with a run-time component and you need to determine an icon ID.

Syntax

- Command Central syntax:

```
cc list resources icons  [options]

 options:
[{--debug | -d}]
[{--error | -r} file]
[{--format | -f} {tsv args | text | xml | csv args | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]
[{--username | -u} user_name]
```

- Not supported on Platform Manager

Arguments and Options

Argument or Option	Description
[options]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To list the icons available for the run-time components managed by the Command Central with host name “rubicon” with port “8090” and have the output returned to the console in XML format:

```
cc list resources icons --server http://rubicon:8090/cce --format xml
--password secret
```

Because the {--username | -u} option is not specified, the command uses the default user name. For more information, see "--username | -u" on page 52. The command specifies “secret” for the user’s password.

Related Commands

cc update inventory components

15 Security Credentials Commands

cc add security credentials

Adds security credentials for an installation or run-time component.

Syntax

- Command Central syntax:

```
cc add security credentials  [nodeAlias=node_alias]
[runtimeComponentId=componentid]  {--input | -i} file{.xml|.json}
[options]
options:
[{--debug | -d}]
[{--error | -r} file]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not supported by Platform Manager

Arguments and Options

Argument or Option	Description
`[nodeAlias=node_alias]`	Optional. Specifies the alias name of the installation for which you want to associate the security credentials. You can view a list of installations and their aliases using cc list landscape nodes.
`[runtimeComponentId=componentid]`	Optional. Specifies the run-time component for which you want to associate the security credentials. You can determine the IDs for run-time components using cc list inventory components.
`{--input | -i} file{.xml|.json}`	Required. Identifies an input file that contains the data for the security credentials. For more information, see "--input | -i" on page 42. **Tip:** To determine how to specify the data in the input file, use cc get security credentials to retrieve data for existing security credentials. For example, if you want to

Argument or Option	Description
	use an XML input file, use cc get security credentials with the `--format xml` option to retrieve the data in XML format.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- By default, if you omit an argument, the command applies the credentials to all items. For example, if you omit the `[runtimeComponentId=componentid]` argument, the command applies the credentials to all run-time components.
- You can set different credentials for the Platform Manager servers in your landscape using the Command Central web user interface or the command line tool. For example, you can configure Command Central to connect to one Platform Manager as “Administrator1/manage1” and the second Platform Manager server as “Administrator2/manage2”. For example:

```
cc add security credentials runtimeComponentId=OSGI-SPM
nodeAlias=sag01 --input c:\inputs\creds_data.xml
```

Note: This command will not return an error. However, Command Central does not use the supplied credentials for connection to specified Platform Manager on the specified installation.

Examples When Executing on Command Central

In the following examples, because the `{--server | -s}` and `{--username | -u}` options are not specified, the commands use the default server and user name. For more information, see "--server | -s" on page 50 and "--username | -u" on page 52. The commands specify “secret” for the user’s password.

- Security credentials data is in the c:\inputs\creds_data.xml file. To add security credentials for all Integration Server run-time components on all installations:

```
cc add security credentials runtimeComponentId=IntegrationServer-instanceName
--input c:\inputs\creds_data.xml --password secret
```

- Security credentials data is in the c:\inputs\creds_data.xml file. To add security credentials for the Integration Server run-time components on installations with alias names “sag01” and “sag02”, use the following two commands:

```
cc add security credentials nodeAlias=sag01
runtimeComponentId=IntegrationServer-instanceName
--input c:\inputs\creds_data.xml --password secret
cc add security credentials nodeAlias=sag02
runtimeComponentId=IntegrationServer-instanceName
--input c:\inputs\creds_data.xml --password secret
```

Related Commands

cc get security credentials

cc delete security credentials

cc delete security credentials

Deletes security credentials from an installation or run-time component.

Syntax

- Command Central syntax:

```
cc delete security credentials  [nodeAlias=node_alias]
[runtimeComponentId=componentid]  [options]
options:
[{--debug | -d}]
[{--error | -r} file]
[--force]
[{--log | -l} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not supported by Platform Manager

Arguments and Options

Argument or Option	Description
[nodeAlias=*node_alias*]	Optional. Specifies the alias name of the installation for which you want to delete security credentials. You can view a list of installations and their aliases using cc list landscape nodes.
[runtimeComponentId=*componentid*]	Optional. Specifies the run-time component for which you want to delete security credentials. You can determine the IDs for run-time components using cc list inventory components.
[*options*]	Optional. Refer to the command syntax for a list of the options the command supports. For a description

Argument or Option	Description
	of the options, see "Options for the Commands" on page 31.

Usage Notes

- By default, if you omit an argument, the command removes the credentials from all items. For example, if you omit the `[runtimeComponentId=componentid]` argument, the command removes the credentials from all run-time components.
- When you remove credentials, Command Central uses the default credentials.

Examples When Executing on Command Central

- To remove the security credentials for the Integration Server run-time component that is running on the installation with alias name “sag02” using the authorization of the user with user name “Administrator” and password “manage”:

  ```
  cc delete security credentials nodeAlias=sag02
  runtimeComponentId=IntegrationServer-instanceName
  --username Administrator --password manage
  ```

  After removing the security credentials, the Integration Server uses the default credentials, that is, user name “Administrator” and password “manage”.

- To remove the security credentials for all Integration Server run-time components running on all installations using the authorization of the user with user name “Administrator” and password “manage”:

  ```
  cc delete security credentials
  runtimeComponentId=IntegrationServer-instanceName
  --username Administrator --password manage
  ```

Related Commands

cc add security credentials

cc get security credentials

cc get security credentials

Retrieves security credentials that are associated with an installation or run-time component.

Syntax

- Command Central syntax:

  ```
  cc get security credentials  [nodeAlias=node_alias]
  [runtimeComponentId=componentid]  [options]
  options:
  [{--accept | -a} content_type]
  [{--debug | -d}]
  [{--error | -r} file]
  ```

```
[{--format | -f} {xml | json}]
[{--log | -l} file]
[{--output | -o} file]
[{--password | -p} password]
[{--quiet | -q}]
[{--server | -s} url]]
[{--username | -u} user_name]
```

- Not supported by Platform Manager

Arguments and Options

Argument or Option	Description
`[nodeAlias=node_alias]`	Optional. Specifies the alias name of the installation for which you want to retrieve security credentials. You can view a list of installations and their aliases using cc list landscape nodes.
`[runtimeComponentId=componentid]`	Optional. Specifies the run-time component for which you want to retrieve security credentials. You can determine the IDs for run-time components using cc list inventory components.
`[options]`	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- If you do not specify the `{--format | -f}` option, the default output format is XML.
- By default, if you omit an argument, the command retrieves the credentials from all items. For example, if you omit the `[runtimeComponentId=componentid]` argument, the command retrieves the credentials for all run-time components.
- For security reasons, the command does not return the password.

Example When Executing on Command Central

To execute a command on the Command Central server with host name “rubicon” and port “8090” to retrieve security credentials for the Integration Server run-time component that is running on the installation with alias name “sag01”, using the

authorization of the user with user name “Administrator” and password “manage”, and have the information displayed on the console in XML format:

```
cc get security credentials nodeAlias=sag01
runtimeComponentId=IntegrationServer-instanceName --format xml
--server http://rubicon:8090/cce --username Administrator
--password manage
```

Related Commands

cc add security credentials

cc delete security credentials

16 Template Commands

cc create templates

Platform Manager creates a new template based on an existing managed installation, archives the new template and transfers the archive file to the Command Central file system.

Syntax

- Command Central syntax:

  ```
  cc create templates {--input | -i} filename{.xml | .json} [options]
  ```

- Platform Manager syntax:

  ```
  cc create templates {--input | -i} filename{.xml | .json} [options]
  ```

Arguments and Options

Argument or Option	Description
`{--input | -i}` *filename{.xml | .json}*	Required. Identifies an input file that contains the template metadata required for the template creation. For more information, see "--input | -i" on page 42.
[*options*]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see"Options for the Commands" on page 31.

Usage Notes

- The template XML metadata required for creating a template should follow the format below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<templateMetadata>

    <!-- template to create / apply -->
    <alias>templateAlias</alias>

    <!-- node alias to create from / apply to -->
    <nodeAlias>nodeAlias</nodeAlias>

    <!-- overwrite template, if exists -->
    <overwriteTemplate>true</overwriteTemplate>

    <!-- capture / install products -->
    <productOption>
        <!-- registered product repository Id: See below -->
        <repositoryId>productRepositoryId</repositoryId>
        <type>PRODUCTS</type>
        <!-- install the latest version available in the above
```

```
            repository -->
        <useLatestVersion>false</useLatestVersion>
    </productOption>

    <!-- capture / install fixes -->
    <fixesOption>
        <!-- registered fix repository id: See below -->
        <repositoryId>fixRepositoryId</repositoryId>
          <!-- install the latest version available in the above
                repository -->
        <useLatestVersion>true</useLatestVersion>
        <type>FIXES</type>
    </fixesOption>

    <!-- capture / copy files referenced from configurations -->
    <filesOption>
        <type>FILES</type>
    </filesOption>

    <!-- capture / apply configuration -->
    <configurationOption>
        <type>CONFIGURATION</type>
    </configurationOption>

</templateMetadata>
```

- When you make configuration changes externally to Command Central, for example by configuring settings in Integration Server Administrator, the changes may not be included in the created template. To ensure that external configuration changes are included in the template, run the `cc list configuration instances` command with the refresh parameter set to `true`.
- To monitor the status of the job scheduled to create the template, specify the job ID returned by the create templates command in the `cc list jobmanager jobs` command. The create templates command also provides a reference to the templates log that you can use to check the logs, using the `cc get diagnostics logs` command.
- A template that is created directly on Platform Manager is not visible to Command Central and cannot be applied to a different node. To apply a template created on Platform Manager, you must manually export the template from the node and import it into Command Central.

Example When Executing on Command Central

To create a template, based on the “templateMetadata.xml” file, on the Command Central server with host name “rubicon” and port “8090”:

```
cc create templates --server http://rubicon:8090/cce -p manage
-i D:\templateMetadata.xml
```

Example When Executing on Platform Manager

To create a template, based on the “templateMetadata.xml” file, on the Command Central server with host name “rubicon” and port “8092”:

```
cc create templates --server http://rubicon:8092/spm -p manage
-i D:\templateMetadata.xml
```

cc delete templates

Removes a template from a Command Central installation.

Syntax

- Command Central syntax:

```
cc delete templates template_alias [options]

 options:
[--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
template_alias	Required. Specifies the template to delete. You can determine the template alias using the `cc list templates` command.
`[`*options*`]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To delete a template in a Command Central installation:

```
cc delete templates sampleTemplate -p manage
```

cc exec templates apply

Applies a template registered and available under the specified alias in a managed installation.

Syntax

- Command Central syntax:

```
cc exec templates apply template_alias
{--input | i}filename{.xml | .json} [options]
```

- Platform Manager syntax:

```
cc exec templates apply template_alias
{--input | i} filename{.xml | .json} [options]
```

Arguments and Options

Argument or Option	Description
template_alias	Specifies the alias for the template to apply. You can determine the template alias using the `cc list templates` command.
`{--input | -i}` *filename{.xml | .json}*	Required. Identifies an input file that contains the template metadata required for applying the template. For more information, see "--input | -i" on page 42.
`[`*options*`]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Usage Notes

- The template XML metadata required for the template application should follow the format below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<templateMetadata>

    <!-- template to create / apply -->
    <alias>templateAlias</alias>

    <!-- node alias to create from / apply to -->
    <nodeAlias>nodeAlias</nodeAlias>

    <!-- overwrite template, if exists -->
    <overwriteTemplate>true</overwriteTemplate>

    <!-- capture / install products -->
    <productOption>
        <!-- registered product repository Id: See below -->
        <repositoryId>productRepositoryId</repositoryId>
        <type>PRODUCTS</type>
        <!-- install the latest version available in the above
             repository -->
        <useLatestVersion>false</useLatestVersion>
    </productOption>

    <!-- capture / install fixes -->
    <fixesOption>
        <!-- registered fix repository id: See below -->
        <repositoryId>fixRepositoryId</repositoryId>
        <!-- install the latest version available in the above
             repository -->
        <useLatestVersion>true</useLatestVersion>
        <type>FIXES</type>
    </fixesOption>
```

```
    <!-- capture / copy files referenced from configurations -->
    <filesOption>
        <type>FILES</type>
    </filesOption>

    <!-- capture / apply configuration -->
    <configurationOption>
        <type>CONFIGURATION</type>
    <replaceConfiguration>false</replaceConfiguration   >
</configurationOption>

</templateMetadata>
```

If `replaceConfiguration` is set to true, the apply template command replaces the existing configuration on the system on which the template is applied with the configuration included in the template. All configurations on the target system that are not part of the applied template will be removed.

- When applying a template with product configuration, you must first install the products and fixes, start the product components, and then apply the template. Generally, the template configuration will be applied successfully on product components that are running.

  Note: Some component configurations require that you restart the component to use the configuration changes. The apply template command does not automatically restart components.

  The result of the apply template command will look as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <landscapeTemplateOperationResultDTO
        xmlns:ns2="http://www.w3.org/2005/Atom">
  <jobId>5</jobId>
  <ns2:link rel="job" href="http://localhost:8090/cce/jobmanager/jobs/5"/>
  <ns2:link rel="log" href="http://localhost:8090/cce/diagnostics/logs/
n1/OSGI-SPM/templates.log/full?search=CREATE1380025427852"/>
    <logName>CREATE1380025427852</logName>
    <nodeAlias>n1</nodeAlias>
    </landscapeTemplateOperationResultDTO>
```

- To monitor the status of the job scheduled to apply the template, specify the job ID returned by the apply templates command in the `cc list jobmanager jobs` command. The apply templates command also provides a reference to the template log that you can use to check the logs, using the `cc get diagnostics logs` command.
- When you apply a template that includes products or fixes, register and configure at least one product and fix repository on the target installation. For information about the register repository commands, see "Repositories Commands" on page 173.

Note: Platform Manager templates are imported from Command Central as part of the apply template command. When you execute the apply template command on Platform Manager, make sure that the template already exists on this Platform Manager.

Example When Executing on Command Central

To apply a template, based on the "templateMetadata.xml" file, on the Command Central server with host name "rubicon" and port "8090":

```
cc exec templates apply --server http://rubicon:8090/cce -p manage -i D:\templateMetadata.xml
```

Example When Executing on Platform Manager

To apply a template, based on the "templateMetadata.xml" file, on the Command Central server with host name "rubicon" and port "8092":

```
cc exec templates apply --server http://rubicon:8092/spm -p manage -i D:\templateMetadata.xml
```

cc export templates

Exports the template available under the specified alias into a compressed file.

Syntax

- Command Central syntax:

  ```
  cc get templates export  template_alias
  {--output | -o} filename.zip [options]
  ```

- Platform Manager syntax:

  ```
  cc get templates export  template_alias
  {--output | -o} filename.zip [options]
  ```

Arguments and Options

Argument or Option	Description
template_alias	Specifies the alias for the template to export. You can determine the template alias using the `cc list templates` command.
`{--output | -o}` *filename*`.zip`	Required. Specifies the output zip archive file to which to export the template. For more information about the `[{--output | -o} file]` option, see "--output | -o" on page 47.
`[`*options*`]`	The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To export a template, under the alias “myAlias”, from the Command Central server with host name “rubicon” and port “8090”:

```
cc get templates export myAlias --server http://rubicon:8090/cce
-p manage --output template-output-file.zip
```

Example When Executing on Platform Manager

To export a template, under the alias “myAlias”, from the Command Central server with host name “rubicon” and port “8092”:

```
cc get templates export myAlias --server http://rubicon:8092/spm
-p manage --output template-output-file.zip
```

cc list templates

Retrieves a list of all templates available in a landscape.

Syntax

- Command Central syntax:

  `cc list templates [options]`

- Platform Manager syntax:

  `cc list templates [options]`

Arguments and Options

Argument or Option	Description
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To list the templates on the Command Central server with host name “rubicon” and port “8090”:

```
cc list templates --server http://rubicon:8090/cce -p manage
```

Example When Executing on Platform Manager

To list the templates on the Command Central server with host name “rubicon” and port “8092”:

```
cc list templates --server http://rubicon:8092/spm -p manage
```

cc exec templates import

Registers an exported template in a Command Central installation.

Syntax

- Command Central syntax:

  ```
  cc exec templates import  {--input | -i} filename.zip [options]
  ```

- Platform Manager syntax:

  ```
  cc exec templates import  {--input | -i} filename.zip [options]
  ```

Arguments and Options

Argument or Option	Description
`{--input | -i} filename.zip`	Required. Identifies an input archive file that contains an exported template. For more information, see "--input | -i" on page 42.
`[options]`	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Options for the Commands" on page 31.

Example When Executing on Command Central

To import a template in a Command Central installation:

```
cc exec templates import --input sampleTemplate.zip -p manage
```

Example When Executing on Platform Manager

To import a template in a Command Central installation:

```
cc exec templates import --input sampleTemplate.zip -p manage
```

17 OSGI Components and the Command Line Interface

Configuration Types that the OSGI Profile Components Supports

The OSGI profile run-time components support creating instances of one or more of the following configuration types:

- COMMON-LOG
- COMMON-PORTS
- COMMON-SYSPROPS
- COMMON-JAAS

The following sections provide more detail about each configuration type and list the products whose OSGI profile supports each configuration type.

COMMON-LOG

Use to configure log levels for log categories and log file locations.

Products whose OSGI profile supports COMMON-LOG
Command Central
Platform Manager
Software AG Runtime

COMMON-PORTS

Use to configure the HTTP, HTTPS, JMX, JDWP (Debug), and/or SSH ports. The following tables lists products that have OSGI profile components that support COMMON-PORTS and the types of ports each support.

Port type	Products whose OSGI profile supports this port type	Description
HTTP	Command Central Platform Manager Software AG Runtime	The HTTP port is enabled by default. You can use the command line interface to add, remove, edit, and validate the HTTP port.
HTTPS	Command Central Platform Manager Software AG Runtime	The HTTPS port is enabled by default. You can use the command line interface to add, remove, edit, and validate the HTTPS port.

Port type	Products whose OSGI profile supports this port type	Description
JMX	Command Central Integration Server My webMethods Server Platform Manager Software AG Runtime	The JMX port is enabled by default. You can use the command line interface to add, remove, and edit the JMX port. You can only define one JMX port.
JDWP (Debug)	Command Central Integration Server My webMethods Server Platform Manager Software AG Runtime	The JDWP (Debug) port is disabled by default. You can use the command line interface to edit this port, but not to remove or add it. This port is used when the run-time component is started in debug mode using the `cc exec lifecycle` command.
SSH	Command Central Integration Server My webMethods Server Platform Manager Software AG Runtime	The SSH port is disabled by default. You can use the command line interface to add, remove, and edit the SSH port. You can only define one SSH port.

COMMON-SYSPROPS

Use to configure the OSGI profile properties defined in the config.ini configuration file:

installation_directory /profiles/*product_code* /configuration/config.ini file

Note: Under normal circumstances, you should *not* modify the config.ini file. It should *only* be modified by the Software AG Installer.

The following tables lists products that have OSGI profile components that support COMMON-SYSPROPS and the *product_code* used in the directory path to the location of the config.ini for that product.

Products whose OSGI profile supports COMMON-SYSPROP	*product_code* used in the directory path to the config.ini file
Command Central	CCE
Integration Server	IS_*instancename*

Products whose OSGI profile supports COMMON-SYSPROP	*product_code* used in the directory path to the config.ini file
	where *instancename* is the name of the Integration Server instance, for example, “IS_default”.
My webMethods Server	MWS_*instancename* where *instancename* is the name of the My webMethods Server instance, for example, “MWS_default”.
Platform Manager	SPM
Software AG Runtime	CTP

COMMON-JAAS

Use to configure profile authentication and authorization configuration:

installation_directory /profiles/*product_code* /configuration/jaas.config file

Note: For the *product_code* in the directory path, refer to the description of COMMON-SYSPROPS.

Products whose OSGI profile supports COMMON-JAAS
Command Central
Platform Manager
Software AG Runtime

For more information, see JAAS file information in the *webMethods Command Central Help*.

Related Commands

cc create configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

Lifecycle Actions for the OSGI Profile Components

The following table lists the actions that the OSGI profile run-time components support with the cc exec lifecycle command and the operations taken against a run-time component when an action is executed.

Action	Products whose OSGI component supports this action	Description
`start`	Command Central Integration Server My webMethods Server Software AG Runtime	Starts the run-time component. When successful, the run-time status is set to ONLINE. When the run-time component starts, the OSGI framework comes online and opens the JMX port. **Note:** To correctly report the ONLINE status, the JMX port must be enabled. **Note:** When using the command line interface, to start Command Central you must execute the command against Platform Manager because the command will fail against Command Central when it is not running.
`startindebugmode`	Command Central My webMethods Server Software AG Runtime	Starts the run-time component in debug mode. When successful, the run-time status is set to ONLINE.
`stop`	Command Central Integration Server My webMethods Server Software AG Runtime	Stops the run-time component. When successful, the run-time status is set to STOPPED.

Action	Products whose OSGI component supports this action	Description
`restart`	Command Central Integration Server My webMethods Server Platform Manager Software AG Runtime	Stops, then restarts the run-time component. The run-time status is set to ONLINE. **Note:** To correctly report the ONLINE status, the JMX port must be enabled.

Related Commands

cc exec lifecycle

Run-time Monitoring Statuses for the OSGI Profile Components

The following table lists the run-time statuses that the OSGI profile run-time components can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The run-time component is running. The run-time component indicates ONLINE when the profile JVM is running and that the JMX port is responding.
FAILED	The run-time component is not running due to some failure, and attempts to start it again have failed.
STARTING	The run-time component is starting.
STOPPED	The run-time component is not running because it was shut down normally.
STOPPING	The run-time component is stopping.
UNKNOWN	The status of the run-time component cannot be determined.
UNRESPONSIVE	The run-time component does not respond to a ping to its JMX port.

Related Commands

cc get monitoring

Run-time Monitoring States for OSGI Profile Components

In response to the `cc get monitoring runtimestate` and `cc get monitoring state` commands, the OSGI profile run-time components provide information about the key performance indicators (KPIs) in the following table.

KPI	Description
JVM memory usage	Use this KPI to monitor the JVM memory usage of the run-time component so that you can take corrective actions if storage use approaches a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is calculated using the following: MAX(Maximum*80%, Maximum-100) This means a marginal value is when there is only 20% free JVM memory available or less than 100MB of JVM memory is available. ■ Critical is calculated using the following: MAX(Maximum*95%, Maximum-50) This means a critical value is when there is only 5% free JVM memory available or less than 50MB of JVM memory left. ■ Maximum amount of memory that is allocated memory for the JVM. **Note:** This KPI value might be incorrect when running in a 32-bit operating systems.
Number of JVM threads	Use this KPI to monitor number of JVM threads that the E run-time component is using so that you can take corrective actions if the number of used threads approaches a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the allocated JVM threads.

KPI	Description
	■ Critical is 95% of the allocated JVM threads. ■ Maximum is calculated using the following: MAX(HWM(value), 500) This means the initial maximum value is 500 threads. However, if the JVM has more than 500 threads, the greater number is used as the maximum.
JVM CPU load	Use this KPI to monitor how much CPU the JVM is using so that you can take corrective actions if the CPU usage approaches a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the CPU usage. ■ Critical is 95% of the CPU usage. ■ Maximum is 100% of the CPU usage. **Note:** This KPI is *only* supported when running on Java 7. It is *not* supported on Java 6. **Note:** This KPI is *not* reported when running on HP-UX.

Related Commands

cc get monitoring

OSGI-CCE-ENGINE Reference

See the following sections:

- "Configuration Types that OSGI-CCE-ENGINE Supports" on page 220
- "Lifecycle Actions for OSGI-CCE-ENGINE" on page 221
- "Run-time Monitoring Statuses for OSGI-CCE-ENGINE" on page 221

Configuration Types that OSGI-CCE-ENGINE Supports

The OSGI-CCE-ENGINE run-time component supports creating instances of the configuration types listed in the following table.

Configuration Type	Use to...
COMMON-LICENSE	Update the Command Central license file.
COMMON-LICLOC	Retrieve the location of the Command Central license file. Updating the license file is not supported.

Related Commands

Configuration Types that the OSGI Profile Components Supports

cc create configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

Lifecycle Actions for OSGI-CCE-ENGINE

The following table lists the actions that the OSGI-CCE-ENGINE run-time component supports with the cc exec lifecycle command and the operation taken against the run-time component when an action is executed.

Action	Description
`restart`	Stops, then restarts the run-time component. When successful, the run-time status is set to ONLINE.

Related Commands

cc exec lifecycle

Run-time Monitoring Statuses for OSGI-CCE-ENGINE

The following table lists the run-time statuses that the OSGI-CCE-ENGINE run-time component can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The run-time component is running.
STOPPED	The run-time component is not running because it was shut down normally.
UNKNOWN	The status of run-time component cannot be determined.

Related Commands

cc get monitoring

OSGI-SPM-ENGINE Reference

See the following sections:

- "Configuration Types that OSGI-SPM-ENGINE Supports" on page 222
- "Run-time Monitoring Statuses for OSGI-SPM-ENGINE" on page 223
- "Run-time Monitoring States for OSGI-SPM-ENGINE" on page 224

Configuration Types that OSGI-SPM-ENGINE Supports

The OSGI-SPM-ENGINE run-time component supports creating instances of the configuration types listed in the following table.

Configuration Type	Use to configure...
COMMON-SYSPROPS	Monitoring service parameters, for example, the products' polling interval for run-time status and date.
SIN-INTERNAL-GROUPS	Internal user groups stored in the common/conf/groups.txt file.
SIN-INTERNAL-ROLES	User roles stored in the common/conf/roles.txt file.
SIN-INTERNAL-USERS	Internal users stored in the common/conf/users.txt file.
SPM-NODEID	Internal unique identifier for a Platform Manager.

Configuration Type	Use to configure...
	Note: Command Central automatically manages unique identifiers. You can customize the identifiers. However, you *must* ensure that each identifier is unique within the landscapes that Command Central manages. To view a list of identifiers already registered with Command Central, use the cc list landscape nodes command.

Related Commands

cc create configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

Run-time Monitoring Statuses for OSGI-SPM-ENGINE

The following table lists the run-time statuses that the OSGI-SPM-ENGINE run-time component can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The run-time component is running.
STOPPED	The run-time component is not running because it was shut down normally.
UNKNOWN	The status of run-time component cannot be determined.

Related Commands

cc get monitoring

Run-time Monitoring States for OSGI-SPM-ENGINE

In response to the `cc get monitoring runtimestate` and `cc get monitoring state` commands, OSGI-SPM-ENGINE run-time component provides information about the key performance indicators (KPIs) in the following table.

KPI	Description
Computer Memory (MB)	Use this KPI to monitor the memory usage of the computer where Platform Manager is running so that you can take corrective actions if memory use approaches a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the maximum amount of physical memory. ■ Critical is 95% of the maximum amount of physical memory. ■ Maximum is the total amount of physical memory.
Computer disk space (MB)	Use this KPI to monitor the available disk space of the computer where Platform Manager is running so that you can take corrective actions if disk space usage approaches a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the maximum amount of physical disk space. ■ Critical is 95% of the maximum amount of physical disk space. ■ Maximum the total amount of physical disk space.
Computer CPU utilization	Use this KPI to monitor the CPU usage of the computer where Platform Manager is running so that you can take corrective actions if CPU usage approaches a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of CPU utilization. ■ Critical is 95% of CPU utilization.

KPI	Description
	■ Maximum 100% of CPU utilization.

Related Commands

cc get monitoring

OSGI-*-TOMCAT-ENGINE Reference

This component runs in multiple profiles, for example:

- OSGI-CTP-TOMCAT-ENGINE for the Software AG Runtime OSGI profile
- OSGI-CCE-TOMCAT-ENGINE for the Command Central OSGI profile
- OSGI-SPM-TOMCAT-ENGINE for the Platform Manager OSGI profile

See the following sections:

- "Lifecycle Actions for OSGI-*-TOMCAT-ENGINE" on page 225
- "Run-time Monitoring Statuses for OSGI-*-TOMCAT-ENGINE" on page 226

Lifecycle Actions for OSGI-*-TOMCAT-ENGINE

The following table lists the run-time statuses that the OSGI-*-TOMCAT-ENGINE run-time component can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Action	Description
`start`	Starts the run-time component. When successful, the run-time status is set to ONLINE. **Important:** Do *not* use the `start` action for either the OSGI-CCE-TOMCAT-ENGINE or OSGI-SPM-TOMCAT-ENGINE run-time components.
`restart`	Stops, then restarts the run-time component. When successful, the run-time status is set to ONLINE.
`stop`	Stops the run-time component. When successful, the run-time status is set to STOPPED. **Important:** Do *not* use the `stop` action for either the OSGI-CCE-TOMCAT-ENGINE or OSGI-SPM-TOMCAT-ENGINE run-time components. Stopping the component ends remote

Action	Description
	communications with the Web user interface and the REST API.

Related Commands

cc exec lifecycle

Run-time Monitoring Statuses for OSGI-*-TOMCAT-ENGINE

The following table lists the run-time statuses that the OSGI-*-TOMCAT-ENGINE run-time component can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The OSGI-*-TOMCAT-ENGINE run-time component is running.
STOPPED	The OSGI-*-TOMCAT-ENGINE run-time component is not running because it was shut down normally.
UNKNOWN	The status of the OSGI-*-TOMCAT-ENGINE run-time component cannot be determined.

Related Commands

cc get monitoring

18 webMethods Broker and the Command Line Interface

Commands that webMethods Broker Supports

webMethods Broker supports the Platform Manager commands listed in the following table. The table lists where you can find general information about each command. Additionally, if there is webMethods Broker-specific information, the table lists where you can learn more about arguments and options that webMethods Broker supports or details about the actions Broker takes when you execute an `exec` command.

Commands	Additional Information
`cc get configuration data`	For general information about the command, see "cc get configuration data" on page 64. For webMethods Broker-specific information about using the command, see "Configuration Types that webMethods Broker Supports" on page 229.
`cc update configuration data`	For general information about the command, see "cc update configuration data" on page 66.
`cc get configuration instances`	For general information about the command, see "cc get configuration instances" on page 70. For webMethods Broker-specific information about using this command, see "Configuration Types that webMethods Broker Supports" on page 229.
`cc list configuration instances`	For general information about the command, see "cc list configuration instances" on page 72. For webMethods Broker-specific information about using this command, see "Configuration Types that webMethods Broker Supports" on page 229.
`cc get configuration types`	For general information about the command, see "cc get configuration types" on page 74. For webMethods Broker-specific information about using this command, see"Configuration Types that webMethods Broker Supports" on page 229.

Commands	Additional Information
`cc list configuration types`	For general information about the command, see "cc list configuration types" on page 76. For webMethods Broker-specific information about using this command, see "Configuration Types that webMethods Broker Supports" on page 229.
`cc get inventory components`	For general information about the command, see "cc get inventory components" on page 98.
`cc list inventory components`	For general information about the command, see "cc list inventory components" on page 99.
`cc exec lifecycle`	For general information about the command, see "cc exec lifecycle" on page 162. For webMethods Broker-specific information about using this command, see "Lifecycle Actions for Broker Server" on page 231.
`cc get monitoring`	For general information about the command, see "cc get monitoring" on page 168. For webMethods Broker-specific information about using this command, see: ■ "Monitoring Run-time Statuses for webMethods Broker" on page 232 ■ "Monitoring Run-time States for webMethods Broker" on page 233

Configuration Types that webMethods Broker Supports

The following table lists the configuration types that webMethods Broker supports.

Configuration Type	Use to configure...
BROKER-MWSADMIN	The protocol, host, and port of the My webMethods Server that hosts the administration user interface for the Broker Server for which the command was executed. The default value is http://localhost:8585.

Configuration Type	Use to configure...
	You can use the following commands to retrieve or update the value: ■ cc get configuration data ■ cc update configuration data
COMMON-ADMINUI	The full URL for the Broker Server Details page in My webMethods. You cannot change the value. You can use the cc get configuration data to retrieve its value.
COMMON-LICENSE	The SagLic license file. You can use the following commands to add and update webMethods Broker-specific information to the SagLic file: ■ cc get configuration data ■ cc update configuration data ■ cc get configuration instances ■ cc list configuration instances ■ cc get configuration types ■ cc list configuration types
COMMON-LICLOC	The location where the license file resides in file system where Broker Server is installed.
COMMON-PORTS	The Broker Server listener port. You can use the following commands to retrieve or update port information: ■ cc get configuration data ■ cc update configuration data **Note:** You cannot use the command line interface to change the Broker Server port number. You can only change the SSL information.

Example When Executing on Command Central

To change the URL of the My webMethods Server that hosts the Broker Server administration user interface to http://localhost:8500, do the following:

1. Read the current configuration of BROKER-MWSADMIN of the Broker-Server-8349 instance and store the configuration details in the BROKER-MWSADMIN.txt file:

```
cc get configuration data node_alias Broker-Server-8349
BROKER-MWSADMIN -p password -o BROKER-MWSADMIN.txt
```

2. Using a text editor, edit the BROKER-MWSADMIN.txt file to change the URL to http://localhost:8500.
3. Update BROKER-MWSADMIN using the new settings in the BROKER-MWSADMIN.txt file:

   ```
   cc update configuration data node_alias Broker-Server-8349
   BROKER-MWSADMIN -p password -i BROKER-MWSADMIN.txt
   ```

4. View the My webMethods Server URL change in the BROKER-MWSADMIN configuration:

   ```
   cc get configuration data node_alias Broker-Server-8349
   BROKER-MWSADMIN -p password -o BROKER-MWSADMIN.txt
   ```

5. Refresh the COMMON-ADMINUI configuration:

   ```
   cc get configuration data node_alias Broker-Server-8349
   COMMON-ADMINUI refresh=true -p password
   ```

Related Commands

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

Lifecycle Actions for Broker Server

The following table lists the actions that webMethods Broker supports with the cc exec lifecycle command and the operation taken against Broker Server when an action is executed.

Action	Description
`start`	Starts Broker Server. When successful, the Broker Server run-time status is set to ONLINE.
`stop`	Stops Broker Server. The Broker Server run-time status is STOPPED.
`restart`	Stops, then restarts Broker Server. The Broker Server run-time status is set to ONLINE.
`pause`	Pauses operation on Broker Server. The Broker Server run-time status is set to PAUSED. When a Broker Server is paused, all Brokers that belong to the Broker Server stop publishing documents. However,

Action	Description
	Broker clients can still retrieve documents from the client queues, and you can still perform administrative tasks, such as creating clients and document types on the paused Brokers.
`resume`	Resumes a previously paused Broker Server. As a result, Brokers that belong to the Broker Server start publishing documents again. The Broker Server run-time status is returns to ONLINE.

Related Commands

cc exec lifecycle

Monitoring Run-time Statuses for webMethods Broker

The following table lists the run-time statuses that webMethods Broker can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	Broker Server is running.
FAILED	Broker Server is not running due to some failure, and attempts to start it again have failed.
PAUSED	All Brokers belonging to the Broker Server have paused document publishing.
STARTING	Broker Server is starting.
STOPPED	Broker Server is not running because it was shut down normally.
STOPPING	Broker Server is stopping.
UNKNOWN	The status of Broker Server cannot be determined.

Run-time Status	Meaning
UNRESPONSIVE	Broker Server does not respond to a ping operation. However, other indicators, such as the existence of the PID or LOCK file indicate that Broker Server is running.

Related Commands

cc get monitoring

Monitoring Run-time States for webMethods Broker

In response to the `cc get monitoring runtimestate` and `cc get monitoring state` commands, webMethods Broker provides information about the following key performance indicators (KPIs):

KPI	Description
Data storage or configuration storage usage	Use this KPI to monitor the Broker Server run-time memory storage or configuration storage so that you can take corrective actions if storage use approaches a critical value. The marginal, critical, and maximum values for this KPI depend on the maximum storage size configured for Broker Server. ■ Marginal is 60% of the maximum storage size. ■ Critical is 80% of the maximum storage size. ■ Maximum is the configured storage size.
Memory usage	Use this KPI to monitor use of Broker Server main memory so that you can take corrective actions if memory use approaches a critical value. The marginal, critical, and maximum values for this KPI depend the amount of main memory configured in the Broker Server configuration file (awbroker.cfg). ■ Marginal is 80% of the maximum main memory. ■ Critical is 95% of the maximum main memory. ■ Maximum is the configured amount of main memory.

KPI	Description

Stalled queues	Use this KPI to monitor stalled queues. A queue is considered stalled only if all the following conditions are true for that queue:
	■ A client is connected to the queue.
	■ The queue contains at least one message.
	■ More than five minutes has elapsed since the last time the client retrieved a message from the queue.
	The KPI uses the following marginal, critical, and maximum values:
	■ Marginal is 1 stalled queue.
	■ Critical is 50% of the maximum number of queues.
	■ Maximum is determined by the greatest value among the following conditions:
	  ■ 1 queue
	  ■ 5% of the total number of client or forward queues in all Brokers.
	  ■ Current number of stalled queues.

For more information about the webMethods Broker KPIs, see information about monitoring webMethods Broker in the *webMethods Command Central Help*.

Related Commands

cc get monitoring

19 CentraSite and the Command Line Interface

Commands that CentraSite Registry Repository Supports

The CentraSite Registry Repository run-time component supports the Platform Manager commands listed in the following table. The table lists where you can find general information about each command. Additionally, the table lists where you can learn more about arguments and options that the run-time component supports or details about the actions i takes when you execute an `exec` command.

Commands	For more information, see...
`cc get inventory components`	For general information about the command, see "cc get inventory components" on page 98.
`cc list inventory components`	For general information about the command, see "cc list inventory components" on page 99.
`cc exec lifecycle`	For general information about the command, see "cc exec lifecycle" on page 162. For CentraSite-specific information about using this command, see "Lifecycle Actions for CentraSite Registry Repository" on page 237.
`cc get monitoring`	For general information about the command, see "cc get monitoring" on page 168. For CentraSite-specific information about using this command, see "Run-time Monitoring Statuses for CentraSite Registry Repository" on page 238.

Commands that CentraSite Application Server Tier Supports

The CentraSite Application Server Tier run-time component supports the Platform Manager commands listed in the following table. The table lists where you can find general information about each command.

Commands	For more information, see...
`cc get inventory components`	For general information about the command, see "cc get inventory components" on page 98.

Commands	For more information, see...
`cc list inventory components`	For general information about the command, see "cc list inventory components" on page 99.

Note: The CentraSite Application Server Tier does not support the `cc exec lifecycle` command. The CentraSite Application Server Tier is a CentraSite component, but it runs inside Software AG Runtime. You cannot start, stop, or restart the CentraSite Application Server Tier independent of the Software AG Runtime (CTP).

Lifecycle Actions for CentraSite Registry Repository

The following table lists the actions that the CentraSite Registry Repository (CRR) run-time component supports with the cc exec lifecycle command and the operation taken against the run-time component when an action is executed.

Action	Description
`start`	Starts the run-time component. When successful, the run-time status is set to ONLINE.
`stop`	Stops the run-time component. When successful, the run-time status is set to STOPPED. **Note:** The CentraSite Server waits for currently active processing to finish before stopping.
`restart`	Stops, then restarts the run-time component. The run-time status is set to ONLINE.
`startindebugmode`	Starts the run-time component in debug mode. When successful, the run-time status is set to ONLINE. While running in Debug mode, CentraSite writes status and other information to log files in the *CentraSite_directory*\data directory.

Related Commands

cc exec lifecycle

Run-time Monitoring Statuses for CentraSite Registry Repository

The following table lists the run-time statuses that the CentraSite Registry Repository (CRR) run-time component can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The run-time component is running.
STARTING	The run-time component is starting.
STOPPED	The run-time component is not running because it was shut down normally.
STOPPING	The run-time component is stopping.
UNKNOWN	The status of the run-time component cannot be determined.

Related Commands

cc get monitoring

20 Integration Server and the Command Line Interface

Commands that Integration Server Supports

Integration Server supports the Platform Manager commands listed in the following table. The table lists where you can find general information about each command. Additionally, if there is Integration Server-specific information, the table lists where you can learn more about arguments and options that Integration Server supports or details about the actions Integration Server takes when you execute an `exec` command.

Commands	Additional Information
`cc create configuration data`	For general information about the command, see "cc create configuration data" on page 59. For Integration Server-specific information about using this command, see "Configuration Types that IntegrationServer-instanceName Supports" on page 242.
`cc delete configuration data`	For general information about the command, see "cc delete configuration data" on page 62.
`cc get configuration data`	For general information about the command, see "cc get configuration data" on page 64.
`cc update configuration data`	For general information about the command, see "cc update configuration data" on page 66.
`cc get configuration instances`	For general information about the command, see "cc get configuration instances" on page 70. For Integration Server-specific information about using this command, see "Configuration Types that IntegrationServer-instanceName Supports" on page 242.
`cc list configuration instances`	For general information about the command, see "cc list configuration instances" on page 72. For Integration Server-specific information about using this command, see "Configuration Types that IntegrationServer-instanceName Supports" on page 242.

Commands	Additional Information
`cc get configuration types`	For general information about the command, see "cc get configuration types" on page 74. For Integration Server-specific information about using this command, see "Configuration Types that IntegrationServer-instanceName Supports" on page 242.
`cc list configuration types`	For general information about the command, see "cc list configuration types" on page 76. For Integration Server-specific information about using this command, see "Configuration Types that IntegrationServer-instanceName Supports" on page 242.
`cc exec configuration validation create`	For general information about the command, see "cc create configuration data" on page 59. For Integration Server-specific information about using this command, see "Configuration Types that IntegrationServer-instanceName Supports" on page 242.
`cc exec configuration validation delete`	For general information about the command, see "cc create configuration data" on page 59.
`cc exec configuration validation update`	For general information about the command, see "cc create configuration data" on page 59. For Integration Server-specific information about using this command, see "Configuration Types that IntegrationServer-instanceName Supports" on page 242.
`cc create instances`	For general information about the command, see "cc create instances" on page 118.
`cc delete instances`	For general information about the command, see "cc delete instances" on page 120. **Important:** You must stop an Integration Server instance before deleting the instance.

Commands	Additional Information
`cc list instances supported products`	For general information about the command, see "cc list instances supported products" on page 121.
`cc update instances`	For general information about the command, see "cc update instances" on page 122.
`cc get inventory components`	For general information about the command, see "cc get inventory components" on page 98.
`cc list inventory components`	For general information about the command, see "cc list inventory components" on page 99.
`cc exec lifecycle`	For general information about the command, see "cc exec lifecycle" on page 162.
`cc get monitoring`	For general information about the command, see "cc get monitoring" on page 168. For Integration Server-specific information about using this command, see: ■ "Run-time Monitoring Statuses for IntegrationServer-instanceName" on page 246 ■ "Run-time Monitoring States for IntegrationServer-instanceName" on page 247

Configuration Types that IntegrationServer-*instanceName* Supports

IntegrationServer-*instanceName* , where *instanceName* is the name of the Integration Server instance, run-time component supports creating instances of the configuration types listed in the following table.

Configuration Type	Use to configure...
COMMON-ADMINUI	Full URL to Integration Server Administrator.

Configuration Type	Use to configure...
COMMON-DBFUNCTION	Configuration instance for database functional aliases for Integration Server.
COMMON-JDBC	Configuration instance for JDBC connection pools for Integration Server.
COMMON-JMS	Configuration instance for JMS settings for Integration Server.
COMMON-JNDI	Configuration instance for JNDI settings for Integration Server.
COMMON-KEYSTORES	Configuration instance for a KeyStore alias that identifies a keystore file or private key within a keystore.
COMMON-LICENSE	License files. IntegrationServer-*instanceName* supports configuration instances for the Integration Server Core and Terracotta license files.
COMMON-LICLOC	Locations where license files reside in file system where Integration Server is installed. IntegrationServer-*instanceName* supports configuration instances for the location of the Integration Server Core license file and for the location of the Terracotta license file.
COMMON-LOGGERS	Configuration instance for Integration Server loggers and server log facilities.
COMMON-PORTS	Ports. IntegrationServer-*instanceName* supports configuration instances for HTTP, HTTPS, FTP, FTPS, and Diagnostics ports.
COMMON-SMTP	Settings for sending e-mail messages.
COMMON-SYSPROPS	Server configuration parameters. IntegrationServer-*instanceName* supports viewing and updating the system configuration parameters defined in the server.cnf configuration file:

Configuration Type	Use to configure...
	Integration Server_directory/instances/*instance_name* /config/server.cnf,where *instance_name* is the name of the Integration Server instance.
COMMON-TRUSTSTORES	Configuration instance for a TrustStore alias that identifies a truststore file.
COMMON-VARS	Configuration instance for global variables. Each variable must be declared using a separate configuration instance.
COMMON-WMMESSAGING	Configuration instance for webMethods Messaging settings for Integration Server. IntegrationServer-*instanceName* supports configuration instances for webMethods Messaging providers (Broker and Universal Messaging).
IS-DEFAULT-WMMESSAGING	The default messaging connection alias.
IS-PRIMARYPORT	The primary port ID for Integration Server.
IS-QUIESCEPORT	The quiesce port ID for Integration Server.
IS-RESOURCES	Configuration instance for resource settings for Integration Server. IntegrationServer-*instanceName* supports configuration instances for outbound HTTP, stateful sessions limit, server thread pool, document stores, and XA recovery store settings.

Related Commands

cc create configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

Lifecycle Actions for Integration Server

The following table lists the actions that webMethods Integration Server supports with the `cc exec lifecycle` command and the operation taken against Integration Server when an action is executed.

Action	Description
`pause`	Switches an active Integration Server to quiesce mode. The Integration Server run-time status is set to PAUSED. When an Integration Server is paused, all ports except the diagnostic port and the quiesce port are disabled. In quiesce mode, any requests that are already in progress are permitted to finish, but any new inbound requests to the server are blocked. Outbound connection attempts, such as connections to JDBC pools or connections through LDAP or a central user directory, remain open.
`resume`	Switches an Integration Server in quiesce mode to active mode and resumes normal operation. All the assets and activities that were disabled or suspended are restored or resumed. The Integration Server run-time status returns to ONLINE.

Integration Server Instance Management

The following table lists the required parameters that you must include when managing Integration Server instances using the Command Central instance management commands:

Command	Parameter	Description
cc create instances	`instance.name=`*`name`*	Required. A name for the new Integration Server instance.
	`primary.port=`*`port`*	Required. The main listening port for the new Integration Server instance.
	`diagnostic.port=`*`port`*	Required. The diagnostic port for the new Integration Server instance.

Command	Parameter	Description
	`jmx.name=`*`port`*	Required. The JMX port for the new Integration Server instance.

Related Commands

cc create instances

Run-time Monitoring Statuses for IntegrationServer-*instanceName*

The following table lists the run-time statuses that the IntegrationServer-*instanceName* run-time component can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The Integration Server is running, and it is accepting and processing requests over the Integration Server primary port.
PAUSED	The Integration Server is in quiesce mode. Integration Server is accepting or processing requests only over the diagnostic port and the quiesce port.
STOPPED	The Integration Server has been stopped. Integration Server is not accepting nor processing requests over the Integration Server primary port.
UNKNOWN	The status of the Integration Server cannot be determined.
UNRESPONSIVE	The Integration Server is running, but not reachable. **Note:** IS-*instanceName* might still report ONLINE, which indicates there is an issue with Integration Server.

Related Commands

cc get monitoring

Run-time Monitoring States for IntegrationServer-*instanceName*

In response to the `cc get monitoring runtimestate` and `cc get monitoring state` commands, IntegrationServer-*instanceName* run-time component provides information about the following key performance indicators (KPIs):

KPI	Description
Running Services	Use this KPI to monitor the number of services that Integration Server is running concurrently so that you can take corrective actions if the number approaches a critical value. The number of running services includes services that were triggered, scheduled, or directly invoked. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the high water mark of concurrently running services. ■ Critical is 95% of the high water mark of concurrently running services. ■ Maximum is 100% of the high water mark of concurrently running services. This is shown in the Threads area of Integration Server.
Response Time	Use this KPI to monitor service response time so that you can take corrective actions if the response time approaches a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the high water mark of service response time. ■ Critical is 95% of the high water mark of service response time. ■ Maximum is 100% of the high water mark of service response time.
Service Errors	Use this KPI to monitor how many service exceptions have occurred in the last minute so that you can take

KPI	Description
	corrective actions if the current number of exceptions is approaching a critical value. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 5 exceptions in the last minute. ■ Critical is 20 exceptions in the last minute. ■ Maximum is more than 20 exceptions in the last minute.

Related Commands

cc get monitoring

21 My webMethods Server and the Command Line Interface

Commands that My webMethods Server Supports

My webMethods Server supports the Platform Manager commands listed in the following table. The table lists where you can find general information about each command. Additionally, the table lists where you can learn more about arguments and options that My webMethods Server supports or details about the actions My webMethods Server takes when you execute an `exec` command.

Commands	For more information, see...
`cc create configuration data`	For general information about the command, see "cc create configuration data" on page 59. For My webMethods Server-specific information about using this command, see "Configuration Types that My webMethods Server-ENGINE Supports" on page 252.
`cc delete configuration data`	For general information about the command, see "cc delete configuration data" on page 62.
`cc get configuration data`	For general information about the command, see "cc get configuration data" on page 64.
`cc update configuration data`	For general information about the command, see "cc update configuration data" on page 66.
`cc get configuration instances`	For general information about the command, see "cc get configuration instances" on page 70. For My webMethods Server-specific information about using this command, see "Configuration Types that My webMethods Server-ENGINE Supports" on page 252.
`cc list configuration instances`	For general information about the command, see "cc list configuration instances" on page 72. For My webMethods Server-specific information about using this command, see "Configuration Types that My webMethods Server-ENGINE Supports" on page 252.

Commands	For more information, see...
`cc get configuration types`	For general information about the command, see "cc get configuration types" on page 74. For My webMethods Server-specific information about using this command, see "Configuration Types that My webMethods Server-ENGINE Supports" on page 252.
`cc list configuration types`	For general information about the command, see "cc list configuration types" on page 76. For My webMethods Server-specific information about using this command, see "Configuration Types that My webMethods Server-ENGINE Supports" on page 252.
`cc exec configuration validation create`	For general information about the command, see "cc create configuration data" on page 59. For My webMethods Server-specific information about using this command, see "Configuration Types that My webMethods Server-ENGINE Supports" on page 252.
`cc exec configuration validation delete`	For general information about the command, see "cc create configuration data" on page 59.
`cc exec configuration validation update`	For general information about the command, see "cc create configuration data" on page 59. For My webMethods Server-specific information about using this command, see "Configuration Types that My webMethods Server-ENGINE Supports" on page 252.
`cc get inventory components`	For general information about the command, see "cc get inventory components" on page 98.
`cc list inventory components`	For general information about the command, see "cc list inventory components" on page 99.
`cc update inventory components`	For general information about the command, see "cc update inventory components" on page 105.

Commands	For more information, see...
`cc exec lifecycle`	For general information about the command, see "cc exec lifecycle" on page 162. For My webMethods Server-specific information about using this command, see"Lifecycle Actions for My webMethods Server-ENGINE" on page 254.
`cc get monitoring`	For general information about the command, see "cc get monitoring" on page 168. For My webMethods Server-specific information about using this command, see: ■ "Run-time Monitoring Statuses for My webMethods Server-ENGINE" on page 254 ■ "Run-time Monitoring States for My webMethods Server" on page 255

Configuration Types that My webMethods Server-ENGINE Supports

My webMethods Server-ENGINE run-time component supports creating instances of the configuration types listed in the following table.

Configuration Type	Use to configure...
COMMON-ADMINUI	Full URL to My webMethods Server. ■ If the My webMethods Server HTTP port is enabled, use the following format where *hostname* is the My webMethods Server host name and *httpport* is the My webMethods Server HTTP port number. http://*hostname*:*httpport* ■ If the My webMethods Server HTTPS port is enabled and the HTTP port is *not* enabled, use the following format where *hostname* is the My webMethods Server host name and *httpsport* is the My webMethods Server HTTPS port number.

Configuration Type	Use to configure...
	https://*hostname* :*httpsport*
COMMON-JDBC	The default connection pool for the My webMethods Server database connection. You can use the command line interface to edit the pool, but not delete it. You can also add, update, or delete additional custom JDBC pools that custom Composite Application Framework (CAF) applications running on My webMethods Server use. **Note:** You can manage instances of this configuration type using the command line interface, but not the Command Central user interface.
COMMON-KEYSTORES	Keystores for My webMethods Server. You can edit the keystores that My webMethods Server uses for its HTTPS port to provide your own keystores. **Note:** You can manage instances of this configuration type using the command line interface, but not the Command Central user interface.
COMMON-PORTS	The My webMethods Server HTTP, HTTPS, and/or AJP13 ports. When adding, editing, and removing ports, keep the following in mind: ■ Ensure at least an HTTP or an HTTPS port is defined. ■ You can only delete the HTTPS port if the HTTP port is defined. If you delete the HTTPS port, you can later add it again. ■ You can only delete the HTTP port if the HTTPS port is defined. If you delete the HTTP port, you can later add it again. ■ There are no restrictions with regards to deleting and/or adding the AJP13 port. ■ There are no restrictions with regards to editing port numbers. **Note:** You can also manage ports in the Command Central user interface if the ports are enabled. After

Configuration Type	Use to configure...
	enabling the ports in My webMethods Server Cluster Administration and restarting My webMethods Server, the ports are visible in the Command Central user interface.
COMMON-SMTP	Settings for sending e-mail messages.
COMMON-TRUSTORES	Truststores for My webMethods Server. You can edit the truststores that My webMethods Server uses for its HTTPS port to provide your own truststores. **Note:** You can manage instances of this configuration type using the command line interface, but not the Command Central user interface.

Related Commands

cc create configuration data

cc get configuration instances

cc list configuration instances

cc get configuration types

cc list configuration types

Lifecycle Actions for My webMethods Server-ENGINE

The My webMethods Server-ENGINE run-time component does not support the use of the cc exec lifecycle command to perform lifecycle actions.

Related Commands

cc exec lifecycle

Run-time Monitoring Statuses for My webMethods Server-ENGINE

The following table lists the run-time statuses that the My webMethods Server-ENGINE run-time component can return in response to the `cc get monitoring runtimestatus` and `cc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Meaning
ONLINE	The My webMethods Server-ENGINE run-time component is running. The run-time component indicates ONLINE when the profile JVM is running and that the My webMethods Server port is responding.
FAILED	The My webMethods Server-ENGINE run-time component is not running due to some failure, and attempts to start it again have failed. The run-time component indicates FAILED when the profile JVM is no longer running, but the Tanuki wrapper PID file still exists.
STARTING	The My webMethods Server-ENGINE run-time component is starting. The run-time component indicates STARTING when the server is starting and reports an HTTP 503, “Not Ready”, status from its HTTP port.
STOPPING	The My webMethods Server-ENGINE run-time component is stopping.
UNKNOWN	The status of The My webMethods Server-ENGINE run-time component cannot be determined.
UNRESPONSIVE	The My webMethods Server-ENGINE run-time component does not respond to a ping to its HTTP/S port.

Related Commands

cc get monitoring

Run-time Monitoring States for My webMethods Server

In response to the `cc get monitoring runtimestate` and `cc get monitoring state` commands, My webMethods Server provides information about the following key performance indicators (KPIs):

KPI	Description
Number of user sessions	Use this KPI to monitor the number of active user sessions so that you can take corrective actions if the number approaches a critical value. ■ Marginal is 80 active user sessions. ■ Critical is 95 active user sessions. ■ Maximum is 100 or more active user sessions.
Number of active connections in the JDBC pool	Use this KPI to monitor the number of active connections in the JDBC pool so that you can take corrective actions if the number of connections approaches a critical value. ■ Marginal is 90 active connections. ■ Critical is 95 active connections. ■ Maximum is 100 or more active connections.
Average response times in milliseconds	Use this KPI to monitor My webMethods Server response times so that you can take corrective actions if the response times slow to a critical value. ■ Marginal is 5000 milliseconds. ■ Critical is 9000 milliseconds. ■ Maximum is response times at 10000 or more millisecond.

Related Commands

cc get monitoring

22 Universal Messaging and the Command Line Interface

Commands that Universal Messaging Supports

Universal Messaging supports the Platform Manager commands listed in the following table. The table lists where you can find general information about each command. Additionally, if there is Universal Messaging-specific information, the table lists where you can learn more about arguments and options that Universal Messaging supports or details about the actions Universal Messaging takes when you execute an `exec` command.

Commands	Additional Information
`cc create configuration data`	For general information about the command, see "cc create configuration data" on page 59. For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc delete configuration data`	For general information about the command, see "cc delete configuration data" on page 62.
`cc get configuration data`	For general information about the command, see "cc get configuration data" on page 64. For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc update configuration data`	For general information about the command, see "cc update configuration data" on page 66.
`cc get configuration instances`	For general information about the command, see "cc get configuration instances" on page 70. For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc list configuration instances`	For general information about the command, see "cc list configuration instances" on page 72.

Commands	Additional Information
	For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc get configuration types`	For general information about the command, see "cc get configuration types" on page 74. For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc list configuration types`	For general information about the command, see "cc list configuration types" on page 76. For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc exec configuration validation create`	For general information about the command, see "cc exec configuration validation create" on page 78. For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc exec configuration validation delete`	For general information about the command, see "cc exec configuration validation delete" on page 81.
`cc exec configuration validation update`	For general information about the command, see "cc exec configuration validation update" on page 83. For Universal Messaging-specific information about using this command, see "Configuration Types That Universal Messaging Supports" on page 261.
`cc get inventory components`	For general information about the command, see "cc get inventory components" on page 98.

Commands	Additional Information
`cc list inventory components`	For general information about the command, see "cc list inventory components" on page 99. For Universal Messaging-specific information about using this command, see "Inventory Information for Universal Messaging" on page 260.
`cc exec lifecycle`	For general information about the command, see "cc exec lifecycle" on page 162. For Universal Messaging-specific information about using this command, see "Lifecycle Actions for Universal Messaging" on page 262.
`cc get monitoring`	For general information about the command, see "cc get monitoring" on page 168. For Universal Messaging-specific information about using this command, see: - "Monitoring Run-time Statuses for Universal Messaging" on page 263 - "Monitoring Run-time States for Universal Messaging" on page 264

Inventory Information for Universal Messaging

The following table lists the information you can retrieve about the Universal Messaging realm servers configured in the *UniversalMessaging_installationDirectory* \nirvana\server directory in an installation. Universal Messaging returns all the folders under the server directory, except the templates.

Property	Value
Display name	Universal Messaging
Run-time component ID	Universal-Messaging-*RealmServerName*
Product ID	NUMRealmServer
Run-time component category	PROCESS

Related Commands

cc get inventory components

cc list inventory components

Configuration Types That Universal Messaging Supports

The following table lists the configuration types that Universal Messaging supports.

Configuration Type	Use to...
COMMON-LICENSE	Configure the SagLic license file. You can use the following commands to add and update Universal Messaging-specific information in the SagLic file: ■ cc get configuration data ■ cc update configuration data ■ cc get configuration instances ■ cc list configuration instances ■ cc get configuration types ■ cc list configuration types
COMMON-LICLOC	View the location of a Universal Messaging realm server's license file. You cannot change the location of the license file.
COMMON-PORTS	Configure the Universal Messaging realm server interfaces. Use the following commands to retrieve, create, update, or delete interface information: ■ cc get configuration data ■ cc create configuration data ■ cc update configuration data ■ cc delete configuration data **Note:** You cannot change the protocol, bind address, port number, or alias of a port of an existing realm server interface.

Configuration Type	Use to...
	Note: If you change the SSL certificates of a secured interface, you must restart the interface.

Examples When Executing on Command Central

- To view the license details of a Universal Messaging realm server instance with "Universal-Messaging-umserver" component ID that runs in the installation with alias name “sag01”:

```
cc get configuration data sag01 Universal-Messaging-umserver
COMMON-LICENSE-Universal-Messaging
```

- To view the license file location of a Universal Messaging realm server instance with "Universal-Messaging-umserver" component ID that runs in the installation with alias name “sag01”:

```
cc get configuration data sag01 Universal-Messaging-umserver
COMMON-LICLOC-Universal-Messaging
```

- To change the license file of a Universal Messaging realm server instance with "Universal-Messaging-umserver" component ID that runs in the installation with alias name “sag01”:

```
cc update configuration data sag01 Universal-Messaging-umserver
COMMON-LICENSE-Universal-Messaging
-i C:\license\nirvana\licensenirvana.xml
```

Related Commands

cc create configuration data

cc delete configuration data

cc list configuration instances

cc get configuration types

cc list configuration types

Lifecycle Actions for Universal Messaging

The following table lists the actions that Universal Messaging supports with the cc exec lifecycle command and the operation taken against a Universal Messaging realm server when an action is executed.

Action	Description
`start`	Starts theUniversal Messaging realm server. When successful, the Universal Messaging realm server run-time status is set to ONLINE.
`stop`	Stops the Universal Messaging realm server. The Universal Messaging realm server run-time status is STOPPED.
`restart`	Stops, then restarts the Universal Messaging realm server. The Universal Messaging realm server run-time status is set to ONLINE.

Related Commands

cc exec lifecycle

Monitoring Run-time Statuses for Universal Messaging

The following table lists the run-time statuses that Universal Messaging can return in response to the `cc get monitoring runtimestatus` command, along with the meaning of each run-time status.

Universal Messaging does not return the STARTING and STOPPING statuses.

Run-time Status	Meaning
ONLINE	Universal Messaging realm server is running.
FAILED	Universal Messaging realm server is not running due to some failure. LOCK file exists.
STOPPED	Universal Messaging realm server is not running because it was shut down normally. LOCK file does not exist.
UNRESPONSIVE	Universal Messaging realm server does not respond to a ping operation. LOCK file exists and the Universal Messaging realm server is running.
UNKNOWN	The status of Universal Messaging realm server cannot be determined.

Related Commands

cc get monitoring

Monitoring Run-time States for Universal Messaging

In response to the `cc get monitoring runtimestate` and `cc get monitoring state` commands, Universal Messaging provides information about the following key performance indicators (KPIs):

KPI	Description
JVM memory usage	Indicates the utilization of JVM memory. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the maximum JVM memory. ■ Critical is 95% of the maximum JVM memory. ■ Maximum is 100% of the maximum JVM memory.
Fanout backlog	Indicates the total number of events currently waiting to be processed by the fanout engine. If the fanout backlog is more than the critical value, there is a possibility that the subscribers receive the published events after some delay. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the maximum value. ■ Critical is 95% of the maximum value. ■ Maximum is 100% of the peak value (high-water mark) of fanout backlog. Default is 100.
Tasks queued for read and write	Indicates the total number of tasks in the read, write, and common read/write pools. If the number of read and write tasks queued is more than the critical value, it indicates that the Universal Messaging realm server is unable to match the speed of the publishers and subscribers. The KPI uses the following marginal, critical, and maximum values: ■ Marginal is 80% of the maximum value.

KPI	Description
	■ Critical is 95% of the maximum value. ■ Maximum is 100% of the peak value (high-water mark) of read and write tasks queued. Default is 100.

For more information about the Universal Messaging KPIs, see information about Universal Messaging monitoring in *webMethods Command Central Help*.

Related Commands

cc get monitoring

A Invoking Commands from Scripts

Creating Shell Scripts that Execute Commands

On Windows, execute `cc` commands within a .bat file execute using `call` statements. The following is an example of a script that might be in a file named get-products-inventory.bat:

```
@echo off
echo getting products inventory
call cc list inventory products
```

Creating Ant Scripts that Execute Commands

You can create Apache Ant scripts that execute Command Central and Platform Manager command line interface commands.

When creating your Ant script, you must:

1. Use the following fragment to declare `cc` Ant tasks:

```
<property environment="os" />
<property="cc.home" value="${os.CC_CLI_HOME}" />

<taskdef resource="com/softwareag/platform/management/client/ant/antlib.xml"
         <classpath>
            <fileset dir="${cc.home}/lib">
                 <include name="*.jar" />
            </fileset>
         </classpath>
</taskdef>
```

2. Create one or more targets that use the `ccsetup` and `cc` tasks. The following shows a sample:

```
<target name="execute-commands-set1" description="Executes cc commands." >
      <ccsetup server="http://localhost:8090/cce"
               username="Administrator"
               password="manage"
               />
      <cc command="list landscape nodes"
          outputFormat="xml"
          />
      <cc ... />
      ...
   </target>
<target name="execute-commands-set2" description="Executes cc commands.">
      <ccsetup server="http://localhost:8092/spm"
               username="Administrator"
               password="manage"
               />
      <cc command="list inventory products"
          outputFormat="json"
          />
      <cc ... />
      ...
   </target>
 ...
```

Parameters to Use with the ccsetup Task

The following table lists the parameters you can use with the `ccsetup` task to set up the base configuration for the script.

Parameter and Description
password Optional. Specifies the password to use for authentication on the Command Central or Platform Manager server. For example: `password="secret"` The following lists the order used to determine the value used for the password: 1. Value set with the `cc` task. 2. Value set with the `ccsetup` task. 3. Value defined in the `CC_PASSWORD` environment variable.
server Optional. Identifies the Command Central or Platform Manager server on which to execute the command. For example: `server="https://localhost:8092/spm"` The following lists the order used to determine the value used for the server: 1. Value set with the `cc` task. 2. Value set with the `ccsetup` task. 3. Value `https://localhost:8090/cce`
username Optional. Specifies the user name to use for authentication. For example: `username="Administrator"` The following lists the order used to determine the value used for the user name: 1. Value set with the `cc` task. 2. Value set with the `ccsetup` task. 3. Value defined in the `CC_USERNAME` environment variable. 4. “Administrator”
trustAllHosts

Parameter and Description
Optional. Specifies whether to trust all hosts. When the parameter is included, Command Central does not verify the name of the server host. For example: `trustAllHosts="true"` The following lists the order used to determine the value for the truststore file: 1. Value set in the custom cc.properties file located in the *user_home* \.sag directory. 2. Value set in the Command Central default cc.properties file located in the `CC_CLI_HOME`\conf directory.
sslTruststoreFile Optional. Specifies the location of the truststore file. For example: `sslTruststoreFile=="${cce.cli.truststore.file.location}"` The following lists the order used to determine the value for the truststore file: 1. Value set in the custom cc.properties file located in the *user_home* \.sag directory. 2. Value set in the Command Central default cc.properties file located in the `CC_CLI_HOME`\conf directory.
sslTruststorePassword Required. Specifies the password for the truststore. `sslTruststorePassword="${cce.cli.truststore.password}"` The following lists the order used to determine the value for the truststore password: 1. Value set in the custom cc.properties file located in the *user_home* \.sag directory. 2. Value set in the Command Central default cc.properties file located in the `CC_CLI_HOME`\conf directory.

Parameters to Use with the cc Task

The following table lists the parameters you can use with the `cc` tasks when executing commands on a Command Central server and/or a Platform Manager server.

Note: Beginning with Command Central and Platform Manager version 9.5.1, Software AG recommends that you use the *inputFormat* and *outputFormat* parameters in place of the *format* , *accept* , and *mediatype* parameters.

Parameter and Description

accept
 	
Deprecated. Optional. Use `outputFormat` in place of `accept`.
 	
Specifies the format for the returned data. You supply a content type with the `accept` parameter that is used on the HTTP Accept request header sent to Command Central or Platform Manager. For example:
 	
`accept="json"`
`accept="xml"`
`accept="csv"`
`accept="tsv"`
 	
If you omit the `accept` parameter, xml is used.
 	
Note: Use either the `accept` or the `format` parameter to specify the format of the returned data. If you specify both, the value you specify with the `accept` is used.

checkevery Optional. Specifies the number of seconds the command waits before checking for expected output specified by the `expectedvalues` parameter. For example: `checkevery="10"` This parameter is only applicable when you also specify the `expectedvalues` parameter. If you specify the `expectedvalues` parameter but omit `checkevery`, the command uses the value of the `CC_CHECK_EVERY` environment variable. If the `CC_CHECK_EVERY` environment variable is not set, the command uses 15 seconds.

command
 	
Optional. Specifies a Command Central or Platform Manager command to execute. For example, to execute the following command:
 	
`cc list landscape nodes`
 	
In a script, use the following:
 	
`<cc command="cc list landscape nodes" />`
 	
Another example might be to execute the following command:
 	
`cc create landscape nodes alias=n1 url=localhost`
 	
In a script, use the following:
 	
`<cc command="cc create landscape nodes alias=n1 url=localhost" />`
 	
Note: Do not include the command options as described in "Options for the Commands" on page 31. Instead use the corresponding attributes listed in this table. For example, if you want to specify the format “json”, use `format="json"` and not `--format json`. In other words, to execute:

Parameter and Description

`cc create landscape nodes alias=n1 url=localhost --format json`
In a script, use the following:
`<cc command="cc create landscape nodes alias=n1 url=localhost" format="json"/>`

debug

Optional. Specifies you want extra information returned that you can use for debugging issues, in addition to the returning service output. The extra information includes:

- HTTP service request
- URL of the Command Central or Platform Manager server to which the request was sent
- Request content type
- Accept header for the request
- HTTP response code from the request
- Response content type
- Response content length

error

Optional. Specifies the file for error output. You can specify:

- Absolute directory path and filename. For example:

  `error="c:\outputs\errors.xml"`
- Relative directory path and filename. For example:

  `error="outputs\errors.json"`
- Filename of a file in the same directory where you initiated the script. For example:

  `output="errors.xml"`

If you omit the `error` parameter, the command output is written to the console.

If you specify both the `error` and the `errorproperty` parameters, the command writes the error output to both locations identified by the parameters.

errorproperty

Optional. Specifies the name of a property where you want error output stored if a command fails and `failonerror="false"`. For example:

`errorproperty="error.property"`

Parameter and Description
If you specify both the `error` and the `errorproperty` parameters, the command writes the error output to both locations identified by the parameters.

expectedvalues

Optional. Specifies the expected values from a command. For example:

```
expectedvalues="STOPPED"
```

Use the `expectedvalues` parameter in conjunction with the `checkevery` and `wait` parameters.

Tip: Using `wait="0"` with `expectedvalues` acts as s simple assertion mechanism to confirm that the output contains what you expect before executing the next step.

If you omit the `expectedvalues` parameter, the command completes without expecting a specific value.

If the expected values that you specify do not match the actual values, the build fails and stops.

failonerror

Optional. Specifies whether to fail the entire script if an error occurs executing the command. Specify:

- `true` if you want the script to fail and stop if an error occurs.
- `false` if you want the script to continue even if the command fails. If the command fails, the error is written to the file specified with the `error` property, the `errorproperty` parameter is set with the command output, and the script can perform additional processing to check the output.

For example:

```
failonerror="false"
```

If you omit the `failonerror` parameter, command uses `true`.

format

Deprecated. Optional. Use `outputFormat` in place of `format`.

Specifies the format you want a command to use for the data it returns. For example:

```
format="xml"
```

Command Central and Platform Manager support the following formats:

- Tab-separated values (`tsv`)
- Plain text (`txt`)
- XML (`xml`)

Parameter and Description

- Comma-separated values (`csv`)
- JavaScript Object Notation (`json`)
- ZIP (`zip`)
- PDF (`pdf`)

If you omit the `format` parameter, the command uses `xml`.

Although Command Central and Platform Manager support these formats, a specific command might only support a subset of the formats. Refer to the documentation for a specific command to determine the exact formats that it supports and to determine the default format for the command.

Note: Not all commands support plain text. If you specify `txt` for a command that does not support this format, the command uses `tsv` or `xml` based on the formats the command supports.

Note: Use either the `accept` or the `format` parameter to specify the format of the returned data. If you specify both, the value you specify with the `accept` is used.

info

Optional. Sets the level of information to log to INFO.

If you omit both the `info` and `quiet` attributes, `info` is used.

input

Required for some actions if `inputstring` is omitted. Identifies a file that contains the input data for the operation. For example, when creating a new installation, you are required to provide an alias name and URL for the installation. You would supply the alias name and URL in the input data file.

When you specify one of the following actions with the `operation` or `method` parameters, specifying input is required. It is not applicable for other actions.

- Operations: `POST`, `CREATE`. `ADD`, `PUT`, `UPDATE`, `EXEC`
- Methods: `POST`, `PUT`

Additionally, specifying input is required when using the `command` parameter if the command you specify requires input.

Supported file types for an input data file are XML (.xml), JavaScript Object Notation (.json), and properties (.properties). Although Command Central and Platform Manager support these formats, a specific command might only support a subset of the formats. Refer to the documentation for a specific command to determine the exact formats that it supports and to determine the default format for the command.

When identifying the input file, you can specify:

Parameter and Description

■ Absolute directory path and filename. For example:
`input="c:\templates\input.xml"`
■ Relative directory path and filename. The path is relative from where you initiated the script. For example:
`input="templates\input.xml"`
■ Filename of a file in the same directory where you initiated the script. For example:
`input="input.xml"`

inputFormat

Optional. Specifies the content type of the input data for a command. You can specify the same values for `inputFormat` and `ouputFormat`.

The default value is taken from the input file extension if the extension matches the short version of a supported media type. If the input file extension does not match the short version of a supported media type, the default is text/plain.

inputstring

Required for some actions if `input` is omitted. Specifies a string that contains the actual input data for the operation.

When you specify one of the following actions with the `operation` or `method` parameters, specifying input is required. It is not applicable for other actions.

- Operations: `POST`, `CREATE`. `ADD`, `PUT`, `UPDATE`, `EXEC`
- Methods: `POST`, `PUT`

Additionally, specifying input is required when using the `command` parameter if the command you specify requires input.

For example, to change the data for the instance with ID “IS-PRIMARYPORT”, for the component with ID “IntegrationServer-*instanceName*”, running on the node with ID “sag01”, you could use the following:

```
<cc command="update configuration data sag01
IntegrationServer-instanceName IS_PRIMARYPORT
inputstring="valid.instance.id" mediaType="text/plain"
format="txt" />
```

Note: Use the `inputstring` attribute when the input data is simple. For more complex data, use the `input` attribute.

log

Optional. Specifies the file for log information. Log information is written whether commands are successful or encounter errors.

Parameter and Description

The logged results include:

- Service output
- Errors that occur while interpreting a command

  Note: If the error occurs while the initializing a command, the error is written to the console rather than the file specified with the `log` parameter

- Debug information if the `debug` parameter is specified

The log information is written to the console if you do not specify the `error` or `output` attributes.

mediatype

Deprecated. Optional. Use `inputFormat` in place of `mediatype`. Specifies the content type of the input data for a command.

method

Required if `operation` is omitted. Use as part of the `command` parameter. Specifies the operation to execute against a resource. For example:

```
method="PUT"
```

Command Central and Platform Manager support the following operations:

- `GET` to retrieve data.
- `POST` to add or create a new resource.
- `PUT` to update data for a resource.
- `DELETE` to delete a data.

If you omit the `method` parameter, you must specify the `operation` parameter to specify the action to execute. Use either the `method` parameter or the `operation` parameter, but not both.

operation

Required if `method` is omitted. Use as part of the `command` parameter. Specifies the operation to execute against a resource. For example:

```
operation="LIST"
```

Command Central and Platform Manager support the following operations:

- `GET` or `LIST` to retrieve data.
- `POST`, `CREATE`, `ADD`, or `EXEC` to add/create a new resource or execute an action against a resource.

Parameter and Description

- `PUT` or `UPDATE` to update data for a resource.
- `DELETE` or `REMOVE` to delete data.
- `OPTIONS` or `WADL` to retrieve information for supported services.

If you omit the `operation` parameter, you must specify the `method` parameter to specify the action to execute. Use either the `method` parameter or the `operation` parameter, but not both. If you specify both, the `operation` parameter is used.

output

Optional. Identifies a file for command output. You can specify:

- Absolute directory path and filename. For example:

  ```
  output="c:\outputs\results.xml"
  ```

- Relative directory path and filename. The path is relative from where you initiated the script. For example:

  ```
  output="outputs\results.json"
  ```

- Filename of a file in the same directory where you initiated the script. For example:

  ```
  output="results.xml"
  ```

If you omit the `output` parameter, the command output is written to the console.

If you specify both the `output` and the `outputproperty` parameters, the command writes the output to both locations identified by the parameters.

outputFormat

Optional. Specifies the format you want a command to use for the data it returns. For example:

```
outputFormat="xml"
```

Command Central and Platform Manager support the following formats:

- Tab-separated values (`tsv`)
- Plain text (`txt`)
- XML (`xml`)
- Comma-separated values (`csv`)
- JavaScript Object Notation (`json`)
- ZIP (`zip`)
- PDF (`pdf`)

The `outputFormat` parameter accepts any value for the HTTP Accept request header sent to Command Central or Platform Manager.

Parameter and Description

If you omit the `outputFormat`, but include an `-o` option in the command, Platform Manager determines the output format from the file extension. If you include a value for the `outputFormat`, for example:

```
cc list landscape nodes -p manage -output-format xml -o D:\f.json
```

Platform Manager uses the `outputFormat` value, in the example the output format will be XML.

If you omit the `outputFormat` parameter and do not include an `-o` option, the command uses `xml`.

Although Command Central and Platform Manager support these formats, a specific command might only support a subset of the formats. Refer to the documentation for a specific command to determine the exact formats that it supports and to determine the default format for the command.

Note: Not all commands support plain text. If you specify `txt` for a command that does not support this format, the command uses `tsv` or `xml` based on the formats the command supports.

outputproperty

Optional. Specifies an ANT property to hold the result of the command. For example:

```
outputproperty="output.property"
```

If you omit the `outputproperty`, the output is written to the console.

If you specify both the `output` and the `outputproperty` parameters, the command writes the output to both locations identified by the parameters.

password

Optional. Specifies the password to use for authentication on the server. For example:

```
password="secret"
```

If you omit the `{--password | -p}` attribute, the command uses the value you specify with the `ccsetup` task. If you do not specify the password with the `ccsetup` task, the command uses the `CC_PASSWORD` environment variable. If the `CC_PASSWORD` environment variable is not set, the build fails indicating the password is not set.

path

Required if `service` and `resource` are omitted. Use as part of the `command` parameter. Specifies a path that identifies the service and resource on which the command acts. To form the path, separate the service and resource by a forward slash (/) or a space. For example:

```
path="inventory/components"
```

Parameter and Description

or

```
path="inventory components"
```

Note: Use either the `path` parameter or the `service` and `resource` parameters to identify the service and resource on which to act.

quiet

Optional. Sets the level of information to log to ERROR.

resource

Required if `path` is omitted. Use as part of the `command` parameter. Specifies the resource against which to execute the command. For example:

```
resource="components"
```

Examples of resources you can supply are:

- `components`
- `environments`
- `fixes`
- `logs`
- `nodes`
- `products`

When you use the `resource` parameter, you must also specify the `service` parameter to identify the service.

Note: Use either the `service` and `resource` parameters or the `path` parameter to identify the service and resource on which to act.

responseCodeProperty

Optional. Specifies an ANT property to hold the response code. For example:

```
responseCodeProperty="response.property"
```

- If a command ends successfully, the property you specify will contain a response code that is 400 or less
- If a command ends with an error and `failonerror` is set to `false`, the property you specify will contain an error code

server

Required if omitted from the `ccsetup` task. Identifies the server on which to execute the command. You can specify either a Command Central or Platform Manager server. For example:

```
server="http://localhost:8092/spm"
```

Parameter and Description
If you omit `server` from the `cc` task, the command uses the value you specify with the `ccsetup` task. If you omit `server` from both tasks, the command uses `http://localhost:8090/cce`.

service

Required if `path` is omitted. Use as part of the `command` parameter. Specifies the service that provides the resource on which the command acts. For example:

```
service="inventory"
```

Examples of services you can supply are:

- `configuration`
- `diagnostics`
- `inventory`
- `jobmanager`
- `landscape`
- `lifecycle`
- `monitoring`
- `resources`

When you use the `service` parameter, you must also specify the `resource` parameter to identify the resource.

Note: Use either the `service` and `resource` parameters or the `path` parameter to identify the service and resource on which to act.

username

Optional. Specifies the user name to use for authentication. For example:

```
username="Administrator"
```

The following lists the order used to determine the value used for the user name:

1. Value set with the `cc` task.
2. Value set with the `ccsetup` task.
3. Value defined in the `CC_USERNAME` environment variable.
4. “Administrator”

wait

Optional. Specifies how many seconds to wait for a long-running operation to return the expected values specified by the `expectedvalues` parameter. For example:

```
wait="160"
```

This parameter is only applicable when you also specify the `expectedvalues` parameter. If you specify the `expectedvalues` parameter but omit `wait`, the

Parameter and Description
command uses the value of the `CC_WAIT` environment variable. If the `CC_WAIT` environment variable is not set, the command uses 120 seconds.